INPE-14798-TDI/1241

ALOCAÇÃO DINÂMICA DE RECURSOS COMPUTACIONAIS PARA EXPERIMENTOS CIENTÍFICOS COM REPLANEJAMENTO AUTOMATIZADO A BORDO DE SATÉLITES

Fabrício de Novaes Kucinskis

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelo Dr. Maurício Gonçalves Vieira Ferreira, aprovada em 30 de março de 2007.

INPE
São José dos Campos
2007

INPE-14798-TDI/1241

ALOCAÇÃO DINÂMICA DE RECURSOS COMPUTACIONAIS PARA EXPERIMENTOS CIENTÍFICOS COM REPLANEJAMENTO AUTOMATIZADO A BORDO DE SATÉLITES

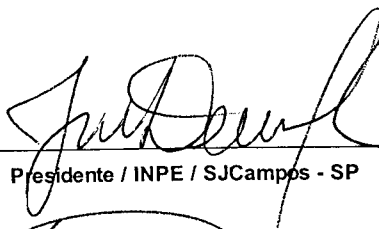
Fabrício de Novaes Kucinskis

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelo Dr. Maurício Gonçalves Vieira Ferreira, aprovada em 30 de março de 2007.

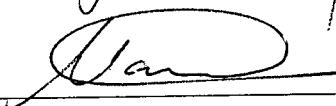
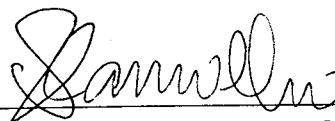
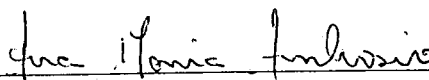
INPE
São José dos Campos
2007

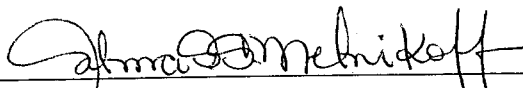
681.3.019

Kucinskis, F. N.

Alocação dinâmica de recursos computacionais para experimentos científicos com replanejamento automatizado a bordo de satélites / Fabrício de Novaes Kucinskis. - São José dos Campos: INPE, 2007.

165 p. ; (INPE-14798-TDI/1241)

1. Inteligência artificial. 2. Planejamento. 3. Escalonamento. 4. Representação do conhecimento. 5. Autonomia. I. Título.

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de Mestre em Computação Aplicada

Dr. José Demisio Simões da Silva — Presidente / INPE / SJCampos - SP

Dr. Mauricio Gonçalves Vieira Ferreira — Orientador(a) / INPE / SJCampos - SP

Dr. Solon Venâncio de Carvalho — Membro da Banca / INPE / SJCampos - SP

Dra. Ana Maria Ambrosio — Membro da Banca / INPE / São José dos Campos - SP

Dra. Selma Shin Shimizu Melnikoff — Convidado(a) / USP / São Paulo - SP

Aluno (a): Fabrício de Novaes Kucinskis

São José dos Campos, 30 de Março de 2007.

“Podemos avistar apenas uma curta distância à frente, mas ali já vemos muito do que precisa ser feito.”

Alan Turing, ao encerrar seu *Computing Machinery and Intelligence*

Às mulheres da minha vida: Olga e Míriam.

AGRADECIMENTOS

Poetas, seresteiros, namorados, correi! É chegada a hora de escrever e cantar, talvez as derradeiras noites de luar.

Com os primeiros versos da letra de *Lunik 9*, Gilberto Gil alertava aos incautos que estava chegando ao fim o tempo de contemplação romântica dos céus. Os soviéticos haviam acabado de chegar à Lua com uma sonda Lunik em 1966, alcançando o que até então era reservado à poesia, filosofia e literatura. O encantamento com o espaço dava lugar à pesquisa.

É exatamente assim que me sinto ao terminar este trabalho. Não raro me surpreendo pela naturalidade com que falo hoje em espaçonaves, órbitas, inteligência artificial; palavras comuns na infância, quando metido com meus brinquedos e minha imaginação, mas que havia deixado de usar quando me tornei 'gente grande'. O encantamento de então deu lugar à pesquisa de hoje. E fico alegre ao constatar que a pesquisa trouxe o encantamento de volta!

Há muitos a quem agradecer pelo apoio e ajuda na realização deste trabalho. Em primeiro lugar, agradeço ao meu orientador Maurício, não apenas pelo ótimo relacionamento e pela orientação segura e tranqüila, pelas dicas mineiras e pela confiança depositada, mas também por ter me apresentado o INPE, alguns anos atrás, e aberto com isso novas portas para mim. Ao Dr. Mário Selingardi, chefe da Divisão de Eletrônica Aeroespacial do INPE, pela oportunidade de estudos que me foi concedida. Ao pessoal do Grupo de Supervisão de Bordo por todo o apoio, pela paciência e pelos esclarecimentos, em especial ao Ronaldo, minha mais freqüente vítima.

Um agradecimento especial à minha esposa Olga, que me deu todo o suporte que precisei, e muitas vezes mais do que mereci, antes e durante a realização deste trabalho. Também à minha mãe Míriam por compreender a ausência dos últimos meses, e por ter criado este seu filho através de bons ensinamentos e de um grande exemplo de vida.

RESUMO

Os experimentos a bordo dos satélites científicos do INPE são atualmente pensados para executar suas tarefas de forma repetitiva, coletando, armazenando e enviando dados em um ciclo que não sofre grandes alterações. Esta forma de lidar com a operação de experimentos é perfeitamente adequada para a observação científica de longo prazo. Existem, entretanto, fenômenos científicos de curta duração cuja ocorrência, embora previsível, é aleatória. Para analisar melhor estes fenômenos, pode ser importante aumentar a taxa de aquisição ou a precisão dos dados coletados. Isso aumenta o consumo de recursos como memória e energia para além do originalmente previsto. Devido à curta duração e à dificuldade em precisar quando um fenômeno deste tipo irá ocorrer, não é suficiente deixar a cargo da equipe de operações em solo a reconfiguração do satélite. O tempo necessário para que o fenômeno seja relatado e para que a equipe de operações crie e envie um novo plano ao satélite é em geral muito maior que a duração do fenômeno. Surge então a necessidade de se permitir que os experimentos, ao detectarem a ocorrência de fenômenos de curta duração, solicitem diretamente ao computador de bordo do satélite a realocação temporária de recursos. Esta realocação deve ocorrer de tal forma que afete o mínimo possível a operação dos outros experimentos e do próprio satélite. Como o número de estados em que o sistema pode estar no momento da detecção do fenômeno é enorme, torna-se difícil o uso de técnicas clássicas de programação para tratá-los. Este trabalho propõe o uso de técnicas de Planejamento e Escalonamento, da área de Inteligência Artificial, para permitir o replanejamento embarcado de operações quando da detecção de fenômenos de curta duração. O objetivo é propiciar ao satélite maior autonomia, e conseqüentemente, maior capacidade de resposta a eventos externos. Foi definida uma arquitetura para um serviço de replanejamento embarcado para os satélites científicos do INPE. Houve a preocupação de se contextualizar esta arquitetura nos projetos atuais do Instituto para satélites e computadores. Foi então desenvolvido um protótipo baseado nesta arquitetura, implementado para execução em um computador de bordo para satélites que está sendo desenvolvido no INPE. Devido à falta de ferramentas de *software* para este tipo de computador, foi preciso desenvolver também uma linguagem de representação do conhecimento e um sistema planejador, específicos para este domínio. O protótipo criado se baseia na idéia de garantir uma maior integração entre o processo de planejamento e o restante do *software* do satélite. A linguagem de representação do conhecimento traz uma forma mais próxima de modelar a operação e o comportamento de satélites do que outras linguagens existentes, não voltadas à área espacial. Os resultados obtidos mostram que o protótipo desenvolvido é adequado para execução no ambiente embarcado, e que este trabalho pode ser considerado um primeiro passo no sentido de aumentar a autonomia do *software* embarcado nos futuros satélites do INPE.

DYNAMIC ALLOCATION OF COMPUTATIONAL RESOURCES FOR SCIENTIFIC EXPERIMENTS WITH AUTONOMOUS REPLANNING ABOARD SATELLITES

ABSTRACT

The experiments aboard the Brazilian scientific satellites are currently thought to execute its tasks in a repetitive way, collecting, storing and sending data in a cycle that does not suffer great alterations. This way of dealing with the experiments operation fits perfectly to long-term scientific observation. There are, however, short-duration scientific phenomena of which occurrence, although predictable, are random. To better analyze these phenomena it may be important to increase the acquisition rate or the precision of the data collected. This increases the consumption of resources, such as memory and power, beyond the originally predicted. Due to the short duration and the difficulty to specify exactly when a phenomenon of this kind will occur, it is not enough to leave the ground operations team in charge of the satellite reconfiguration. The necessary time for the phenomenon to be reported and for the ground team to create and send a new operation plan to the satellite is in general much longer than the duration of the phenomenon. There is then the need for allowing the experiments, when detecting the occurrence of a short-duration phenomenon, to request from the onboard computer the temporary reallocation of resources. This reallocation shall occur in a way that affects the least possible the operation of the other experiments and the satellite itself. As the number of states in which the system can be is huge, it becomes difficult the use of classical programming techniques to handle it. This work proposes the use of Artificial Intelligence Planning and Scheduling techniques to allow the onboard replanning of operations, when a short-duration scientific phenomenon is detected. The main goal is to provide more autonomy to the satellite and, consequently, more strength to respond to external events. It was defined an architecture for an onboard replanning service, to be used in INPE's scientific satellites. There was the concern for context this architecture in the current INPE's projects for satellites and computers. Thus, it was developed a prototype based on this architecture, implemented for execution in a satellite onboard computer which is being developed at INPE. Due to the lack of software tools for this kind of computer, it was also necessary to develop a knowledge representation language and a planning system, specific for this domain. The prototype created is based on the idea of guaranteeing a greater integration between the planning process and the rest of the satellite software. The knowledge representation language brings a form of modeling closer to the operation and behavior of satellites than other existing languages, which are not directed to the space area. The results gotten show that the prototype is adequate for execution in the onboard environment, and that this work can be considered a first step in the direction of increasing the autonomy of the software aboard future INPE's satellites.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

3CS	-	*Three Corner Sat*
ABSTRIPS	-	*Abstraction-Based STRIPS*
ADL	-	*Action Description Language*
AEB	-	Agência Espacial Brasileira
AIPS	-	*International Conference on Artificial Intelligence Planning Systems*
AML	-	*ASPEN Modeling Language*
AOMPS	-	*Autonomous On-board Mission Planning Software*
ASE	-	*Autonomous Sciencecraft Experiment*
ASPEN	-	*Automated Scheduling and Planning Environment*
BIRD	-	*Bispectral InfraRed Detection*
CASPER	-	*Continuous Activity Scheduling, Planning, Execution and Replanning*
CAST	-	*Chinese Academy of Space Technology*
CBERS	-	*China Brazil Earth Resources Satellites*
CCSDS	-	*Consultative Committee for Space Data Systems*
CLARAty	-	*Coupled Layer Architecture for Robotic Autonomy*
CNES	-	*Centre National d'Etudes Spatiales*
COMAV	-	Computador Avançado
CSP	-	*Constraint Satisfaction Problem*
DCAPS	-	*Data Chaser Automated Planning System*
DEA	-	Divisão de Eletrônica Aeroespacial

DLR - *Deutsches Zentrum für Luft-und Raumfahrt*

DS-1 - *Deep Space One*

ECP - *European Conference on Planning*

EO-1 - *Earth Observing One*

EQUARS - *Equatorial Atmosphere Research Satellite*

ESA - *European Space Agency*

EUROPA - *Extensible Universal Remote Operations Planning Architecture*

EUVE - *Extreme Ultraviolet Explorer*

FBM - *French-Brazilian Microsatellite*

FNC - Fórmula Normal Conjuntiva

GCC - *GNU Cross Compiler*

GPL - *GNU General Public License*

GPSS - *Ground Processing Scheduling System*

HSP - *Heuristic Search Planner*

HSTS - *Heuristic Scheduling Testbed System*

IA - Inteligência Artificial

ICAPS - *International Conference on Automated Planning and Scheduling*

INPE - Instituto Nacional de Pesquisas Espaciais

IPC - *International Planning Competition*

JPL - *Jet Propulsion Laboratory*

MAGA - *Multi-Agent Ground-operation Automation*

MAPGEN - *Mixed-initiative Activity Plan GENerator*

MECB - Missão Espacial Completa Brasileira

MER - *Mars Exploration Rovers*

MIRAX - Monitor e Imageador de Raios-X

MSL - *Mars Science Laboratory*

NASA - *National Aeronautics and Space Administration*

OAR - *Online Applications Research Corporation*

OCL - *Object-Centered Language*

OMPS - *Open Multi-CSP Planner and Scheduler*

OO - Orientação a Objetos

PDDL - *Planning Domain Definition Language*

PICo - *Planning Incrementally for Contingencies*

PlanIPOV - Planejamento Inteligente de Planos de Operação de Vôo

PNAE - Programa Nacional de Atividades Espaciais

PO - Pesquisa Operacional

PST - *Planning and Scheduling Team*

RASSO - *Resources Allocation Service for Scientific Opportunities*

RASSO_ml - *RASSO modeling language*

RAX - *Remote Agent Experiment*

RAX-PS - *RAX Planner / Scheduler*

RTEMS - *Real-Time Executive for Multiprocessor Systems*

SACI - Satélite de Aplicações Científicas

SCD - Satélite de Coleta de Dados

SCE - Programa de Satélites Científicos e Experimentos

SNLP - *Systematic Nonlinear Planning*

SPIKE - *Science Planning Intelligent Knowledge Environment*

SPSS - *Science Planning and Scheduling System*

STRIPS - *Stanford Research Institute Problem Solver*

STScI - *Space Telescope Science Institute*

SUBORD - Grupo de Supervisão de Bordo

UCPOP - *Universal Conditional Partial Order Planner*

UFO-1 - *UHF Follow On One*

CAPÍTULO 1

INTRODUÇÃO

O Instituto Nacional de Pesquisas Espaciais (INPE) vem desenvolvendo a tecnologia de satélites há quase três décadas, desde a aprovação da Missão Espacial Completa Brasileira (INPE, 1999) em 1979. Dividida em três grandes ramos, a MECB previa o desenvolvimento de pequenos satélites de aplicações, de um veículo lançador compatível com estes satélites, e a implantação de toda a infra-estrutura necessária para estes projetos.

O processo de desenvolvimento tecnológico então iniciado culminou no bem sucedido lançamento do primeiro satélite brasileiro, o Satélite de Coleta de Dados (SCD-1), em fevereiro de 1993. O êxito alcançado pelo SCD-1 e posteriormente pelo SCD-2, ambos tendo ultrapassado em muito a vida útil estimada, trouxe ao Brasil o reconhecimento de sua capacitação na área espacial.

Visando aumentar esta capacitação com missões mais complexas e de maior valor agregado, o INPE iniciou um programa de parceria internacional com a *Chinese Academy of Space Technology* (CAST) para o desenvolvimento de satélites avançados de sensoriamento remoto, chamados *China-Brazil Earth Resources Satellites* (INPE, 2003). Hoje, os dados e imagens gerados pelos satélites CBERS-1 e 2 são consumidos diariamente por centenas de usuários, nas mais diversas áreas de atuação, dentro e fora do país. Outros três satélites da série CBERS encontram-se atualmente em etapas diversas de desenvolvimento.

Além dos programas de satélites de coleta de dados e de sensoriamento remoto, o INPE possui também o programa de Satélites Científicos e Experimentos (INPE, 2006), do qual fizeram parte os Satélites de Aplicações Científicas (SACI-1 e 2) e o *French-Brazilian Microsatellite* (FBM), em parceria com o *Centre National d'Etudes Spatiales* (CNES) francês. O objetivo destas missões é prover acesso freqüente e de baixo custo

ao espaço para a comunidade científica brasileira, bem como serem plataformas para testes de novas tecnologias e equipamentos para uso espacial, que possam ser futuramente incorporadas em programas de maior vulto, como é o caso do CBERS. Os próximos satélites previstos para o SCE são o *Equatorial Atmosphere Research Satellite* (EQUARS) e o Monitor e Imageador de Raios-X (MIRAX), ambos atualmente em desenvolvimento.

Os satélites do programa SCE possuem como carga útil um conjunto de experimentos científicos e tecnológicos, cujo controle se baseia em modos de operação pré-definidos e na execução de telecomandos temporizados. Estes são comandos gerados e validados em solo, que são então enviados para o satélite e agendados para execução futura. Ao conjunto de telecomandos temporizados é dado o nome de 'plano de operações'.

Qualquer anomalia relacionada à operação dos experimentos é relatada pelo satélite para a equipe de operação em solo via telemetria. Após analisar os dados sobre a anomalia e sobre o estado do satélite, a equipe (junto aos responsáveis pelos experimentos e outros envolvidos na missão) pode ou não modificar o plano de operações corrente. Como a comunicação ocorre apenas quando o satélite passa sobre sua estação terrena de rastreio, o tempo transcorrido entre a detecção da anomalia e a recepção e execução de um novo plano de operações para lidar com ela pode ser de dezenas de horas.

Esta forma de operação dos experimentos é perfeitamente adequada para observações científicas de longa duração. Existem, entretanto, fenômenos de curta duração cuja ocorrência é aleatória – uma perturbação ionosférica, por exemplo, pode ocorrer a qualquer momento e se manifestar por poucos minutos, ou por algumas horas. Para analisar melhor estes fenômenos, pode ser importante aumentar a taxa de aquisição ou a precisão dos dados coletados por um experimento. Isso aumenta o consumo de recursos como energia e memória de armazenamento para além do originalmente previsto.

Devido à curta duração e à dificuldade em precisar quando um fenômeno deste tipo irá ocorrer, não é suficiente deixar a cargo da equipe de operações em solo a geração de um novo plano de operações que reconfigure o sistema. O tempo necessário para que o

fenômeno seja relatado e para que a equipe de operações gere e envie um novo plano ao satélite é em geral muito maior que a duração do fenômeno. Neste caso, a oportunidade científica para analisá-lo adequadamente terá sido perdida.

Surge então a necessidade de se permitir que os experimentos, ao detectarem a ocorrência de fenômenos de curta duração, solicitem diretamente ao computador de bordo do satélite a modificação do plano de operações corrente de forma a realocar temporariamente os recursos necessários para que sejam capazes de realizar uma análise mais detalhada. Entretanto, esta realocação deve ocorrer de tal forma que afete o mínimo possível a operação dos outros experimentos e do próprio satélite. Como o número de estados em que o sistema pode estar no momento da detecção do fenômeno é enorme, torna-se difícil o uso de técnicas clássicas de programação para tratá-los.

Neste contexto, as técnicas de Planejamento e Escalonamento da área de Inteligência Artificial (IA), apresentam-se como uma solução em potencial a ser explorada, e uma das mais promissoras tecnologias aptas a aumentar a autonomia dos satélites. O Planejamento tem como base a simulação da execução de ações sobre determinado ambiente e a avaliação do efeito destas ações sobre o mesmo, de forma que possa ser encontrado um conjunto de ações (um plano) que leve o ambiente de um estado inicial a um estado desejado, ou estado-objetivo. O Escalonamento diz respeito à atribuição de tempo e recursos para cada ação, de forma a satisfazer restrições impostas pelo ambiente. Estes conceitos serão descritos em maiores detalhes no próximo Capítulo.

Mas, apesar de seu poder, a implementação de replanejamento embarcado em satélites encontra dois grandes obstáculos: a capacidade de processamento limitada dos computadores de bordo e a resistência de gerentes de missão e engenheiros com relação ao aumento da autonomia. Esta resistência é natural e até mesmo justificada, uma vez que o custo e o volume de trabalho de uma missão espacial são em geral considerados muito grandes para confiar ao satélite a tomada de mais decisões do que as de rotina, como a correção de atitude e órbita.

O desafio reside em permitir o aumento da autonomia sem, entretanto, reduzir a confiança no comportamento do satélite, e em adequar a aplicação de Planejamento e Escalonamento ao poder de processamento disponível no computador de bordo.

1.1 Motivação do Trabalho de Pesquisa

Ao longo dos anos, vem aumentando a necessidade de que os sistemas espaciais dependam cada vez menos da intervenção de operadores em solo, e tornem-se mais autônomos. Os motivos para isso variam desde a necessidade de redução do tempo de resposta de espaçonaves a eventos externos, até a dificuldade em manter equipes de operadores qualificados disponíveis em tempo integral, o que encarece a operação das missões.

Entretanto, o aumento da autonomia traz como custo uma demanda por maior poder computacional embarcado. Apesar de ser tentador pensar que problemas de desempenho do *software* embarcado serão resolvidos em um futuro próximo por processadores mais rápidos, isso nem sempre é verdade. Uma análise mais cuidadosa mostra que conforme o poder de processamento aumenta, também aumenta a demanda por seu consumo, devido à implementação de novas aplicações e de melhorias nas aplicações existentes.

Outro problema com relação ao aumento da autonomia vem da crença de que todas as decisões tomadas pelos sistemas embarcados em satélites devem ser absolutamente previsíveis. Saber exatamente como um sistema irá se comportar em cada uma das situações que lhe forem apresentadas aumenta a segurança com relação à sua robustez, mas reduz o número de situações às quais o sistema poderá gerar uma resposta adequada.

A escolha deste trabalho de pesquisa foi motivada pela necessidade de se conciliar o aumento da autonomia de satélites com as restrições de poder computacional embarcado, garantindo ainda um nível aceitável de confiabilidade e previsibilidade quanto às respostas do sistema autônomo. É certo que, uma vez superada a dificuldade inicial em se desenvolver, validar e implementar sistemas com maior autonomia, seus

benefícios serão consideráveis: será possível haver uma melhoria na execução de tarefas rotineiras, a execução de novas aplicações não previstas atualmente, e uma potencial redução de custos na operação de satélites.

A tecnologia escolhida para este estudo do aumento da autonomia em satélites é a de Planejamento e Escalonamento em IA, que vem sendo estudada e aplicada gradualmente no controle de missões espaciais, notadamente da NASA e da ESA. Embora a maioria dos projetos seja para a geração de planos de operação em solo, há casos relatados de planejamento embarcado em espaçonaves da NASA.

1.2 Objetivos do Trabalho de Pesquisa

Este trabalho de pesquisa visa o estudo e a definição de uma arquitetura que permita a alteração do plano de operações (ou replanejamento) de um satélite científico, quando da detecção da ocorrência de um fenômeno científico em órbita, de forma a reconfigurar o satélite para fazer uma melhor observação do fenômeno. Deve ser desenvolvido um protótipo que implemente as funcionalidades da arquitetura, total ou parcialmente. Foi escolhido um problema que, devido às suas características de ocorrência aleatória e curta duração, dificilmente seria resolvido sem o replanejamento embarcado.

É necessário ainda que o protótipo forneça meios de aumentar gradativamente a confiança do pessoal da missão com relação ao incremento da autonomia. Para isso, ele deve possuir um conjunto de características que visem auxiliar o processo de aceitação desta tecnologia em missões reais:

- Alto nível de integração ao restante do *software* embarcado – o replanejamento deve ser tratado como apenas mais um serviço provido pelo computador de bordo, tal qual o processamento de telemetria ou tarefas de *housekeeping* e diagnose;

- Modos de execução que permitam uma implantação gradual do sistema, baseada no aumento da confiança com relação aos resultados obtidos;

- Uma forma de representação do conhecimento que facilite o entendimento, por parte dos engenheiros de missão, do processo de decisão do sistema, mesmo que eles não possam prever com exatidão a resposta gerada para cada situação apresentada ao sistema.

1.3 Metodologia de Trabalho e Organização Desta Dissertação

Este trabalho foi iniciado com o estudo da forma de operação atual de satélites do INPE, em especial dos satélites científicos. O resultado deste estudo é descrito no Capítulo 2 desta Dissertação.

A seguir foi realizado um levantamento da bibliografia existente na área de Planejamento e Escalonamento em IA, e um estudo de diversos sistemas planejadores e das técnicas que eles utilizam. Uma visão geral de planejamento e sistemas planejadores é apresentada no Capítulo 3.

Foram então estudados sistemas planejadores que aplicam IA, utilizados em missões espaciais. Foram analisados tanto sistemas planejadores executados em solo quanto planejadores embarcados em espaçonaves. O Capítulo 4 descreve brevemente estes sistemas, com suas principais características, missões em que foram aplicados, seus objetivos e resultados obtidos.

O próximo passo foi analisar a estrutura dos *softwares* embarcados em satélites do INPE e determinar onde um sistema de replanejamento se encaixaria, e como se comunicaria com o restante do *software* embarcado. Isso, aliado a uma análise da arquitetura dos poucos sistemas planejadores embarcados existentes, permitiu a definição de uma arquitetura para replanejamento dos satélites do INPE. Esta arquitetura e os conceitos relacionados a ela são apresentados no Capítulo 5.

Com uma arquitetura definida, passou-se à implementação de um protótipo, que envolveu também o desenvolvimento de uma linguagem de representação do conhecimento própria para o domínio de satélites e de um sistema planejador específico. O Capítulo 6 detalha a etapa de implementação.

Para validar o protótipo foi criado um cenário de testes, composto por um modelo do satélite, planos de operação e previsões de passagens para algumas órbitas, que simula a operação de um satélite científico e dispara o processo de replanejamento embarcado. Este cenário e os resultados obtidos de sua execução são apresentados no Capítulo 7.

O Capítulo 8 traz as conclusões, contribuições, e trabalhos futuros vislumbrados.

CAPÍTULO 2
CONTROLE E OPERAÇÃO DE SATÉLITES CIENTÍFICOS

Neste Capítulo são apresentados os conceitos de missão espacial, sua divisão em segmento espacial e segmento solo, e a forma de comunicação entre os segmentos. A seguir é descrita a forma atual de operação de experimentos científicos. O Capítulo encerra apresentando os procedimentos de resposta à detecção de fenômenos científicos em órbita, tanto na abordagem atual, baseada em decisões tomadas pela equipe de operações, quanto numa abordagem autônoma, proposta nesta Dissertação.

2.1 Definição de Missão Espacial, Sistema Espacial e Seus Segmentos

De acordo com Wertz e Larson (1999), uma missão espacial é o esforço para desenvolver e operar um sistema espacial com um objetivo específico. Este objetivo pode ser o fornecimento de imagens de solo obtidas do espaço, a realização de experimentos científicos em ambientes de microgravidade ou a retransmissão de dados e voz em sistemas de telecomunicação. Um sistema espacial é dividido em dois segmentos: o segmento espacial e o segmento solo.

O segmento espacial consiste da espaçonave em si (no caso do INPE, do satélite; outros exemplos são sondas, *space shuttles*, etc.), que por sua vez é composta por plataforma e carga útil. Ainda conforme as definições de Wertz e Larson, a plataforma de um satélite possui equipamentos que dão suporte à operação das cargas úteis, tais como estrutura, energia, computador de bordo e outros. É a plataforma a responsável pelo monitoramento, operação e manutenção do satélite em órbita, e pela comunicação com as estações terrenas de rastreio e com a carga útil.

A carga útil é a parte dedicada à aplicação, ou seja, ao motivo pelo qual o satélite foi lançado. É ela que gera os dados da missão. Em um satélite de sensoriamento remoto, por exemplo, a carga útil é composta por câmeras e outros sensores. Em um satélite científico, por experimentos científicos e tecnológicos.

O segmento solo é composto pelas estações terrenas, bem como por toda a infra-estrutura e pessoal envolvidos nas operações de monitoramento e manutenção do satélite, e de disponibilização dos produtos do satélite (imagens, dados científicos, etc.) aos seus usuários. A Figura 2.1 ilustra os segmentos espacial, solo, os usuários da missão e suas interações.

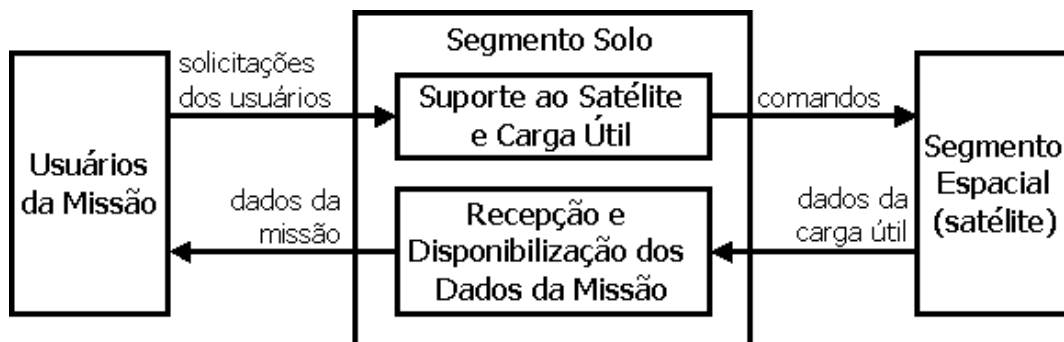



FIGURA 2.1 - Interação entre os Segmentos Solo e Espacial e os Usuários da Missão

Fonte: adaptada de Wertz e Larson (1999, p. 623)

2.2 Pacotes de Telemetria e de Telecomandos

O monitoramento, o controle e a comunicação com o satélite são feitos através de sistemas de telemetria e de telecomandos.Um sistema de telemetria tem como propósito transportar, de forma transparente e confiável, informações de medição de uma origem de dados remotamente localizada para usuários distantes desta origem (CCSDS, 1987-a). Um sistema de telecomandos deve transportar informações de controle de um ponto de origem (como um operador humano) para um equipamento remotamente localizado (CCSDS, 1987-b). As unidades de telemetria e telecomandos são chamadas de pacotes, ou mensagens.

Os pacotes de telemetria são enviados do satélite para a estação terrena. Eles podem conter dados sobre o estado do satélite, como temperatura, níveis de tensão elétrica, situação de seus subsistemas (se estão ligados ou não, seus modos de operação e parâmetros de funcionamento atuais), ou dados de carga útil, como dados brutos de imagens, valores lidos por sensores de experimentos, etc. Pacotes de telemetria

carregam ainda relatos sobre o sucesso ou falha na execução de comandos, mensagens de erro e alertas sobre anomalias detectadas.

Durante sua órbita, o satélite só pode receber comandos e transmitir dados durante o período em que há 'visibilidade' entre sua antena e a da estação terrena, ou seja, no período em que é possível a comunicação entre as antenas de solo e de bordo. Isso se dá quando ele se encontra sobre a estação terrena. A este período de visibilidade e comunicação entre o satélite e a estação terrena é dado o nome de 'passagem'. O tempo de passagem varia de uma órbita para outra, podendo mesmo não ocorrer em alguns casos.

Nos satélites operados pelo INPE, todos categorizados como de órbita baixa (os SCDs orbitam a cerca de 750 km de altitude, e os CBERS a 778 km), e com duração média de cerca de 100 minutos, o tempo médio de passagem gira em torno de 10 minutos, ou seja, 10% do seu tempo em órbita. Para lidar com esta limitação e monitorar o satélite durante toda a sua órbita, existem dois tipos de telemetria: a telemetria de tempo real e a armazenada.

A telemetria de tempo real é utilizada para monitorar, durante o período da passagem, o status dos principais subsistemas do satélite, como a bateria e o computador de bordo. Assim, caso seja detectada qualquer anomalia, comandos podem ser enviados para tentar corrigi-la ainda durante a passagem.

A telemetria armazenada pode conter dados sobre os estados dos equipamentos, relatos da execução de comandos, alertas, mensagens de erro e dados da carga útil. Estes dados são gerados durante a órbita e armazenados para envio em passagens futuras.

Os pacotes de telecomandos são enviados da estação terrena para o satélite. Eles contêm comandos gerados pela equipe de operações, para serem executados pelo satélite. Estes comandos podem ser para ligar ou desligar equipamentos, mudar modos de operação de subsistemas, carregar novos parâmetros de funcionamento, etc. Devido ao tempo limitado de comunicação com o satélite, é necessário prover meios de executar

comandos a qualquer momento durante sua órbita. Para isso, existem dois tipos de telecomandos: os imediatos e os temporizados.

Os telecomandos imediatos devem ser executados assim que são recebidos pelo satélite. Exemplos destes comandos são a sincronização do relógio interno do satélite com o de solo, o carregamento de novos programas, ou comandos emergenciais para a correção de anomalias ou recuperação de modos de erro.

Os telecomandos temporizados contêm um campo com uma referência temporal, indicando ao *software* do satélite quando eles devem ser executados. Ao serem recebidos, eles são armazenados numa estrutura de dados chamada 'fila de telecomandos temporizados'. Esta fila é verificada regularmente e os comandos lá contidos são encaminhados para execução nos seus devidos momentos. O conjunto de telecomandos temporizados armazenados nesta fila compõe o plano de operações corrente do satélite.

2.3 Modos de Operação de Experimentos Científicos

Os experimentos científicos podem ter como objetivo a observação de fenômenos internos ou externos ao satélite. No primeiro caso, eles realizam análises de seus objetos de estudo em ambiente contido fisicamente dentro do experimento, geralmente para a pesquisa do comportamento, em ambiente de microgravidade, de fenômenos conhecidos e bem estudados na presença de gravidade. No segundo caso, os experimentos carregam sensores que coletam dados sobre o ambiente espacial no qual o satélite se encontra. Alguns destes sensores podem detectar a passagem de partículas pelo satélite, outros podem efetuar medidas diversas das camadas superiores da atmosfera, por exemplo.

A concepção atual dos experimentos no programa SCE do INPE é o de 'experimentos inteligentes'. Estes experimentos possuem seu próprio processador, responsável pelo funcionamento do experimento e pela comunicação com o computador de bordo. Entretanto, apesar de possuírem capacidade de processamento própria, os experimentos

compartilham outros recursos providos pelo satélite, como memória para armazenamento de dados e energia.

Os experimentos científicos têm seus modos de operação determinados ainda na etapa de projeto do satélite. Os modos de operação são definidos em termos das fases da órbita em que eles ficarão ligados, do tempo em que permanecerão ligados, de qual a quantidade de dados que irão gerar e quanto irão consumir de energia, entre outros. As fases da órbita podem ser o período em que o satélite encontra-se iluminado pelo Sol, o período em que se encontra eclipsado pela Terra (grosso modo, os períodos de 'dia' e 'noite', durante a órbita), ou mesmo períodos de tempo não relacionados à incidência solar.

Existem diversos fatores que restringem os experimentos. Entre eles, destacam-se a quantidade de recursos disponíveis no satélite (o quanto uma bateria pode fornecer de energia, por exemplo), as restrições de comunicação (taxas de transmissão de telemetria, duração e freqüência das passagens sobre a estação terrena), e as restrições impostas pelas características orbitais, como a exposição de experimentos à luz solar – objetivas de câmeras e espelhos de telescópios, por exemplo, não devem ser expostos diretamente a ela.

Finalmente, há um processo de negociação entre os engenheiros da missão e os cientistas responsáveis pelos experimentos, que serão seus usuários. Como os recursos disponíveis são limitados, cada quantidade de recurso alocado a um experimento significa menos recursos disponíveis aos demais. O resultado deste processo de negociação é um conjunto de quotas de recursos definidas para cada experimento.

Uma vez lançado o satélite, seus experimentos passam a operar da forma previamente definida. Entretanto, as prioridades dos experimentos podem mudar durante a missão, e também as quotas de recursos associadas a eles. Por exemplo, determinado experimento voltado à análise de partículas emitidas pelo Sol pode dispor de maior quantidade de recursos durante um período onde é prevista maior atividade solar.

Como o aumento das quotas para um experimento novamente implica na redução das quotas dos demais, estas alterações devem ser previstas com antecedência suficiente para permitir a análise de seu impacto sobre o funcionamento do satélite, e, em alguns casos, uma nova rodada de negociações entre os cientistas.

A Figura 2.2 ilustra o processo normal de planejamento da operação de satélites científicos no INPE.

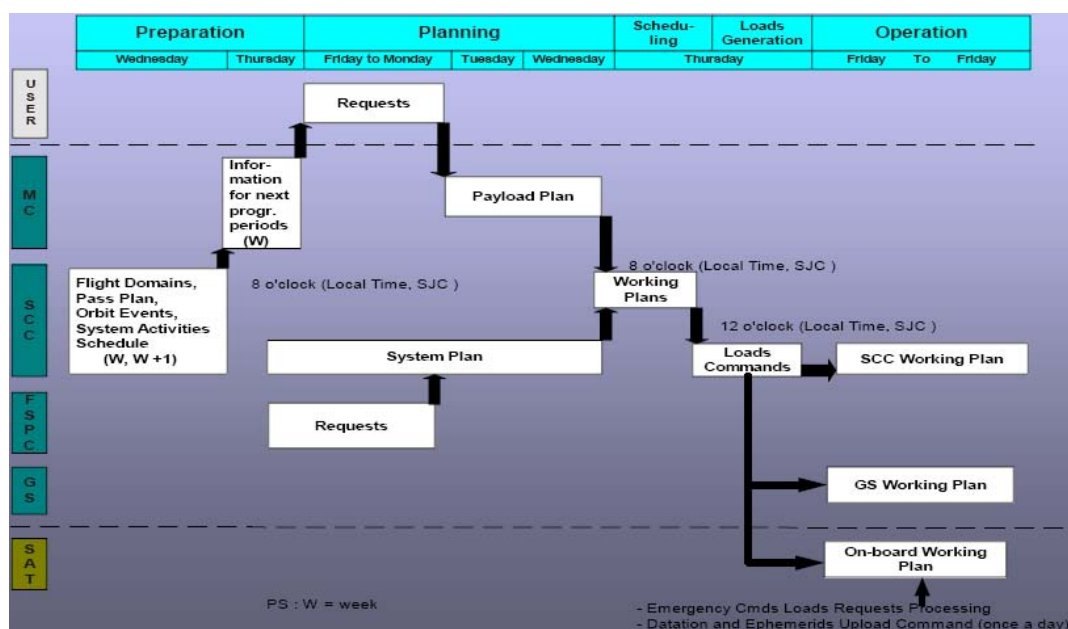



FIGURA 2.2 - O Processo Normal de Planejamento de Satélites Científicos do INPE

Fonte: Carvalho (2001)

O processo normal de planejamento se inicia com o levantamento, por parte do centro de controle de satélites (SCC, na Figura 2.2) e do controle da missão (MC) de informações sobre os domínios de vôo, plano de passagens, eventos orbitais, e o escalonamento de atividades do sistema.

São então obtidas as solicitações dos usuários da missão (que, no caso dos satélites científicos, são os cientistas responsáveis pelos experimentos) e dos responsáveis pela operação do satélite, gerando dois planos: um para a carga útil (os experimentos), e

outro para a plataforma. Estes dois planos são então unidos em um plano operacional, que é enviado ao satélite e colocado em execução.

Nota-se na Figura 2.2 que o processo normal de planejamento, iniciado numa quarta-feira, resulta em um plano operacional que é colocado em execução apenas na sexta-feira da semana seguinte.

Embora o processo normal de planejamento não seja o mesmo que seria seguido no caso da necessidade de se modificar o plano de operações em resposta à detecção de fenômenos científicos de curta duração, ele dá uma boa idéia do tempo despendido e do volume de trabalho envolvido no planejamento de uma missão espacial.

2.4 Resposta à Detecção de Fenômenos Científicos de Curta Duração

A forma de operação dos experimentos descrita no item anterior é perfeitamente adequada ao controle de experimentos de longa duração, ou seja, aqueles que coletam informações sobre determinado objeto de estudo por longos períodos de tempo e de forma repetitiva. Entretanto, existem fenômenos de interesse científico (que serão chamados aqui apenas de 'fenômenos científicos') cuja ocorrência e duração são aleatórias – uma perturbação ionosférica, por exemplo, pode ocorrer a qualquer momento e se manifestar por poucos minutos, ou por algumas horas.

Um experimento que detecte este tipo de fenômeno pode precisar aumentar sua taxa de aquisição ou a precisão dos dados coletados para fazer uma melhor observação, ou pode precisar simplesmente manter-se operando por mais alguns minutos, caso o seu desligamento esteja próximo de ocorrer. Em qualquer um destes casos, ele irá precisar de mais recursos do que as quotas que lhe foram originalmente alocadas.

Seguindo a forma de operação atual, tudo o que o experimento pode fazer ao detectar o fenômeno é disparar um alerta para solo relatando a ocorrência, e seguir operando em seu modo normal até que a equipe de operações em solo envie uma resposta ao alerta que reconfigure o satélite apropriadamente para sua observação.

Um sistema capaz de reconfigurar o satélite sem interferência de solo poderia dar uma resposta imediata à detecção. A Figura 2.3 ilustra as etapas, da detecção à sua resposta, tanto da abordagem atual, dependente das decisões do pessoal da missão, quanto de uma abordagem autônoma, proposta nesta Dissertação. Cada etapa é identificada na figura, e descrita a seguir.

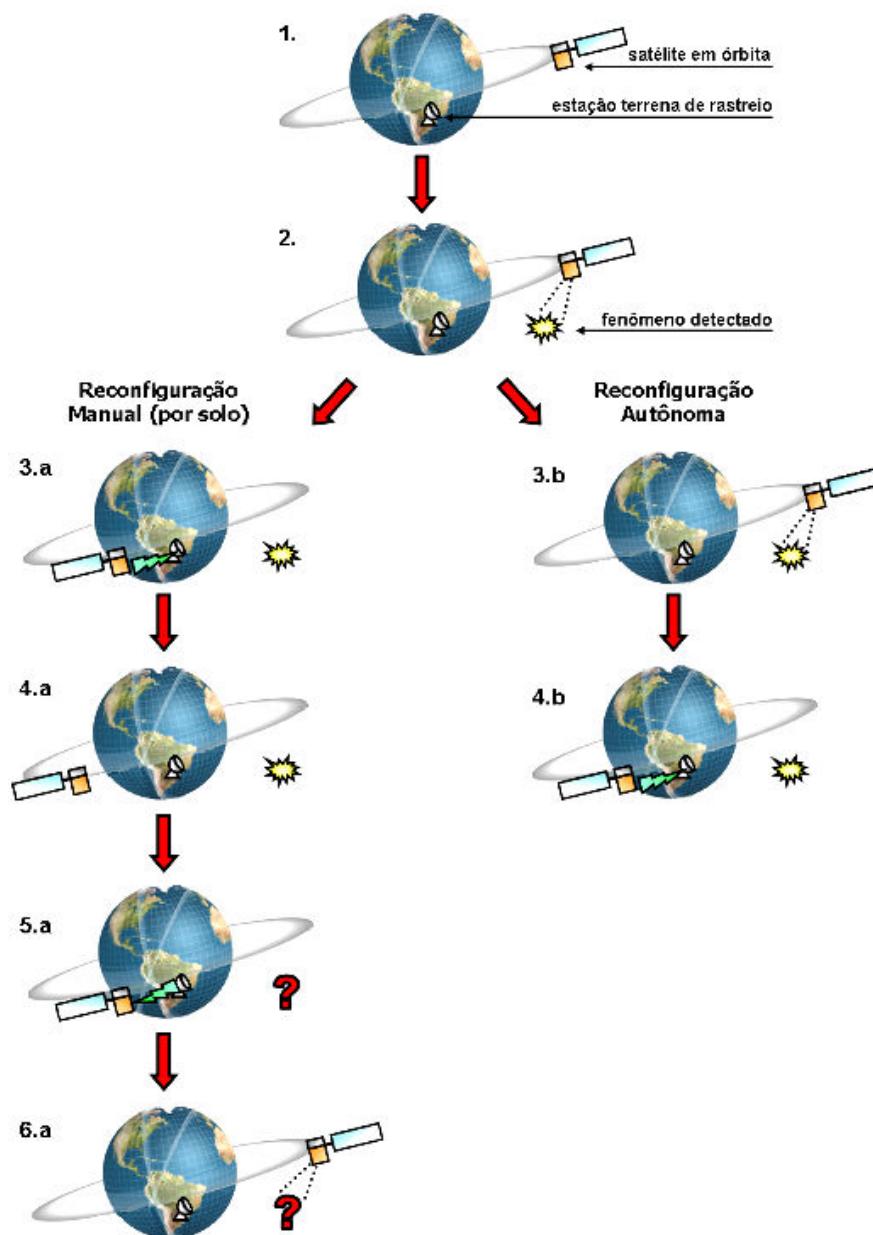



FIGURA 2.3 - Diferentes Respostas à Detecção de Fenômenos Científicos

Segue uma descrição detalhada de ambas as abordagens:

1. O satélite encontra-se em órbita, operando conforme o programado (em modo 'normal' de operações);

2. Um de seus experimentos detecta a ocorrência de um fenômeno científico de curta duração, de interesse para a missão;

Na abordagem atual, o procedimento é o seguinte:

3.a. O experimento gera um alerta sobre a detecção. Na próxima passagem sobre a estação terrena (ou em alguma outra passagem futura), este alerta é enviado para a equipe de operações;

4.a. O satélite segue operando da forma previamente estabelecida enquanto o pessoal da missão encaminha o alerta para o cientista responsável de forma rotineira, junto aos dados de seu experimento. Informado da ocorrência, o cientista entra em contato com o pessoal de missão e solicita modificações no modo de operação de seu experimento (maior tempo ligado, mais memória para armazenar dados coletados, etc.). Esta solicitação será analisada e, caso seja aceita, será criado um novo plano de operações, com telecomandos para alterar a configuração do satélite;

5.a. O novo plano é então enviado para o satélite e colocado em execução;

6.a. O satélite, já reconfigurado, inicia as observações com maior quantidade de recursos dedicados ao experimento que fez a detecção inicial. Entretanto, devido ao tempo transcorrido desde o início do processo, é possível que o fenômeno não esteja mais se manifestando.

Nota-se que este processo pode levar dias, e é seguro supor que o cientista, ao receber o alerta da detecção do fenômeno, irá apenas lamentar a oportunidade perdida. Um sistema autônomo embarcado no satélite, com a capacidade de efetuar a reconfiguração necessária, estaria em condições de dar uma resposta satisfatória. Trabalhando dentro de

uma ‘margem de manobra’ aceitável e por tempo limitado, ele poderia propiciar uma observação adequada e devolver o satélite ao seu modo de operações normal, antes mesmo de enviar um alerta para solo. Este procedimento autônomo, também apresentado na Figura 2.3, é descrito a seguir:

3.b. Após a detecção, o experimento envia uma mensagem ao computador de bordo, solicitando mais recursos para a observação, e informando por quanto tempo irá precisar destes recursos. Um *software* de replanejamento autônomo efetua diversas simulações, analisando diferentes mudanças na configuração do satélite a partir de seu estado atual, e escolhe a que menos afete a operação dos demais experimentos. Um novo plano é gerado em bordo e colocado em execução. Já operando pelo novo plano, o satélite realoca os recursos solicitados e os mantém assim pelo tempo necessário à observação. Ao término deste período, os recursos são devolvidos aos experimentos (e outros subsistemas) que os cederam, e o satélite volta a operar conforme o originalmente programado;

4.b. Numa próxima passagem sobre a estação terrena, o satélite envia o alerta da detecção do evento, junto aos dados coletados durante sua observação.

A tecnologia escolhida para propiciar ao satélite o replanejamento autônomo de sua operação é a de Planejamento e Escalonamento em IA. O próximo Capítulo apresenta uma visão geral desta tecnologia.

CAPÍTULO 3

PLANEJAMENTO E ESCALONAMENTO EM INTELIGÊNCIA ARTIFICIAL

De acordo com Fukunaga et al. (1997), planejamento em IA é "a seleção e sequenciamento de atividades de forma que elas atinjam um ou mais objetivos, e satisfaçam um conjunto de restrições do domínio". Escalonamento é definido por Zweben et al. (1993) como "o processo de atribuir tempo e recursos a tarefas em um plano, satisfazendo ainda uma série de restrições de domínio".

Uma definição menos formal, mas certamente mais esclarecedora, foi dada por Myers e Smith (1999): "por 'planejamento' nós geralmente nos referimos ao processo de decidir o que fazer; [...] nós usamos o termo 'escalonamento' geralmente para designar o processo de decidir quando e como fazer".

Este Capítulo apresenta os conceitos de planejamento e escalonamento, um breve histórico das áreas e as principais abordagens para a solução de problemas de cada uma delas. É então mostrado como estas técnicas são complementares, e como elas vêm sendo integradas nos últimos anos.

3.1 Conceitos de Planejamento

A área de pesquisa em planejamento se originou de uma melhor estruturação das técnicas clássicas de IA de busca pela solução de problemas e da união destas técnicas com os sistemas baseados em conhecimento, que atualmente são baseados no conceito de agentes. Segundo Russell e Norvig (2004), um agente é um sistema computacional com a capacidade de sentir seu ambiente, tomar decisões a respeito do que fazer baseado nos dados ambientais, e agir.

Um agente de solução de problemas é capaz de considerar os efeitos de seqüências de ações antes de agir. Já um agente baseado em conhecimento pode selecionar ações

baseadas em representações lógicas e explícitas do estado corrente e dos efeitos das ações. Um sistema planejador une estes dois tipos de agentes em um agente de planejamento. Ainda de acordo com Russell e Norvig, um agente de planejamento difere de um agente de solução de problemas na sua forma de representação de estados, objetivos, e ações.

A idéia básica do processo de planejamento é simular o comportamento de um ambiente ao qual é aplicada uma seqüência de ações (também chamadas na literatura existente de tarefas, atividades ou operadores). As características do ambiente são representadas de forma lógica por um conjunto de estados, e cada ação aplicada ao ambiente pode modificar um ou mais de seus estados. A meta é encontrar a seqüência correta de ações que leve o ambiente de seu estado inicial a um estado desejado, ou estado-objetivo.

Para isso o planejador deve possuir um modelo do ambiente com representações de seus estados, um conjunto de ações aplicáveis ao modelo – cada uma possuindo suas pré-condições para execução e a descrição de seus efeitos – e os estados inicial e objetivo.

O processo de planejamento começa quando, a partir de um estado inicial, o sistema planejador escolhe, entre um conjunto de ações aplicáveis, qual a que parece levar o modelo para um estado mais próximo do objetivo. As pré-condições para a execução da ação são avaliadas e, caso sejam verdadeiras, os efeitos da ação são aplicados sobre o modelo. É verificado então o novo estado obtido.

Este processo segue iterativamente, até que o estado-objetivo seja atingido. À seqüência de ações que obteve sucesso em levar o modelo de seu estado inicial para o estado-objetivo é dado o nome de 'plano'.

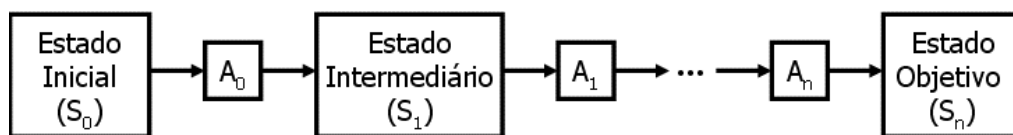



FIGURA 3.1 - Aplicação de Ações no Modelo Até a Obtenção do Estado-Objetivo

Existem diversas técnicas para guiar o processo de busca até uma seqüência de ações adequada. Estas técnicas vêm evoluindo com o passar dos anos e tornando o planejamento em IA capaz de resolver problemas cada vez mais complexos. Os próximos itens apresentam um breve histórico da evolução do planejamento em IA e de algumas das principais abordagens para a solução de problemas de planejamento.

3.2 Histórico e Abordagens de Planejamento

3.2.1 O Primeiro Planejador e Seus Sucessores

O primeiro *software* planejador foi o STRIPS (*Stanford Research Institute Problem Solver*), criado no início da década de 1970 (Fikes e Nilsson, 1971). O STRIPS trouxe um esquema de representação de estados e ações baseado em literais, também chamadas de predicados.

Em STRIPS, uma ação é definida em função dos operadores de pré-condição (PRE), de adição (ADD) e de eliminação (DEL). Os operadores PRE são um conjunto de literais que, se estiverem presentes no estado atual, permitem a aplicação da ação ao estado. Os operadores ADD são um conjunto de literais que são adicionadas ao próximo estado, após a aplicação da ação ao estado atual. Finalmente, os operadores DEL são um conjunto de literais que devem ser removidas no próximo estado, após a aplicação da ação ao estado atual.

Além da representação das ações, STRIPS descreve o problema como conjuntos de predicados que definem seus estados inicial e objetivo. A Figura 3.2 traz um exemplo da representação em STRIPS para o problema do mundo dos blocos, que é definido a seguir.

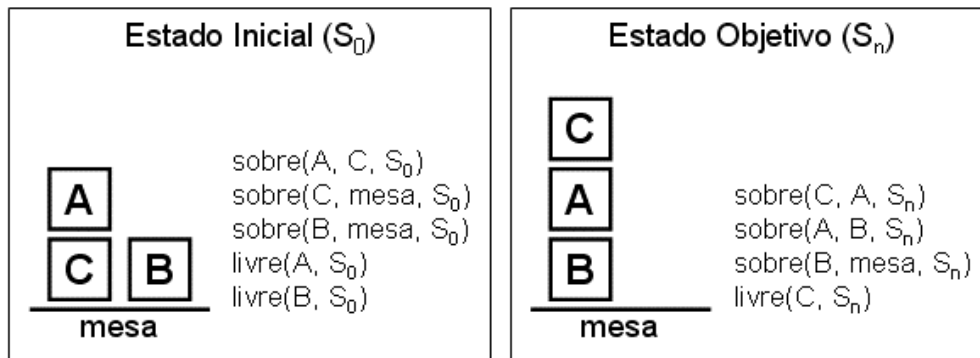



FIGURA 3.2 - Descrição de um Problema em STRIPS

O problema do mundo dos blocos é um problema clássico da área de IA, e será utilizado como exemplo em alguns pontos deste trabalho. Para o problema, assume-se uma mesa com blocos identificados de forma única, empilhados uns sobre os outros num determinado número de colunas. O objetivo do problema é rearranjar os blocos de forma a se obter determinada configuração. Para atingir este objetivo, as seguintes regras devem ser respeitadas:

1. Deve-se mover apenas um bloco por vez;
2. Podem ser movidos apenas blocos que não estejam sob nenhum outro bloco.

A definição de uma ação aplicável a este problema, em STRIPS, é:

```
mover (bloco, de, para)
      PRE: sobre(bloco, de), livre(bloco), livre(para)
      ADD: sobre(bloco, para), livre(de)
      DEL: sobre(bloco, de), livre(para)
```

O algoritmo do STRIPS seleciona um dos literais do objetivo e aplica seqüências de ações ao estado inicial até atingir um estado que contenha este literal. Então seleciona o próximo literal do objetivo e repete o processo, até encontrar um estado que contenha todos os literais do estado objetivo.

Diversos outros planejadores foram baseados nos conceitos do STRIPS, dentre os quais podem ser destacados:

- O ABSTRIPS (Sacerdoti, 1974), que trouxe a primeira abordagem de planejamento hierárquico, adicionando prioridades aos operadores de pré-condição das ações do STRIPS. Estas prioridades eram usadas para gerar um esboço de plano (uma abstração) que resolvia todas as pré-condições de maior nível. Na seqüência, as prioridades do nível imediatamente inferior eram resolvidas, e o processo seguia até a obtenção de um plano com todas as pré-condições resolvidas;

- O PRODIGY (Minton et al., 1989), que adicionou a regressão encadeada de estados, podendo assim realizar uma busca em profundidade em ambos os sentidos (do estado inicial ao objetivo e vice-versa);

- O SNLP (Mcallester e Rosenblitt, 1991), que definiu um plano como um conjunto de ações parcialmente ordenadas, através do uso de restrições de ordenação entre as ações e *links* causais;

- E o UCPOP (Penberthy e Weld, 1992), uma extensão do SNLP que trouxe operadores condicionais e implementou satisfação de restrições para provar a consistência dos planos obtidos.

Mas o grande legado do STRIPS foi a sua forma de descrição de estados e ações, que é a base para as principais formas de representação do conhecimento em planejamento criadas até hoje.

3.2.2 As Conferências na Área de Planejamento

Durante as décadas de 1970 e 1980 a área de planejamento em IA experimentou uma evolução lenta, mas contínua. Cada novo planejador adicionava novos conceitos e ferramentas à implementação original do STRIPS e isso tornava possível sua aplicação a problemas maiores, em domínios mais complexos.

O aumento do poder do planejamento atraiu um maior interesse para a área, o que levou à realização, em 1990, da *International Conference on Expert Planning Systems* que, dois anos depois, se tornou a bienal *International Conference on Artificial Intelligence Planning Systems* (AIPS).

A partir de 1998, passou a ocorrer dentro da AIPS a *International Planning Competition* (IPC). O objetivo desta competição é promover o desenvolvimento de sistemas de planejamento avançados e incentivar a pesquisa competitiva. Na IPC, os planejadores são aplicados a problemas previamente definidos pelos organizadores da competição e comparados em termos do número de problemas resolvidos, do tempo total para a solução dos problemas e do tamanho dos planos gerados.

A AIPS e a IPC trouxeram avanços significativos para a pesquisa em planejamento, como uma maior troca de experiências entre grupos de pesquisa e a criação da linguagem de representação de domínios *Planning Domain Definition Language* (PDDL), além de terem aumentado ainda mais o interesse de pesquisadores de IA pela área. Prova disso foi a união do AIPS com a *European Conference on Planning* (EPC, que vinha sendo realizada também bienalmente desde 1991), o que resultou na *International Conference on Automated Planning and Scheduling* (ICAPS), um evento ainda maior, de realização anual a partir de 2003.

3.2.3 Planejamento Baseado em Satisfatibilidade

Satisfatibilidade Proposicional (SAT) é o problema de se determinar, para uma dada expressão booleana composta por ‘n’ elementos (variáveis), se existe algum conjunto de atribuições de verdadeiro e falso para os elementos que torne a expressão verdadeira (Cook e Mitchell, 1997).

Proposto por Kautz e Selman (1992), o planejamento baseado em satisfatibilidade, implementado originalmente no planejador SATPLAN, trouxe a formalização de problemas de planejamento como conjuntos de cláusulas proposicionais. A satisfação deste conjunto de cláusulas por um algoritmo chamado GSAT apresentou um

desempenho superior ao dos demais planejadores da época. A Figura 3.3 apresenta as etapas do processo de planejamento baseado em Satisfatibilidade Proposicional.

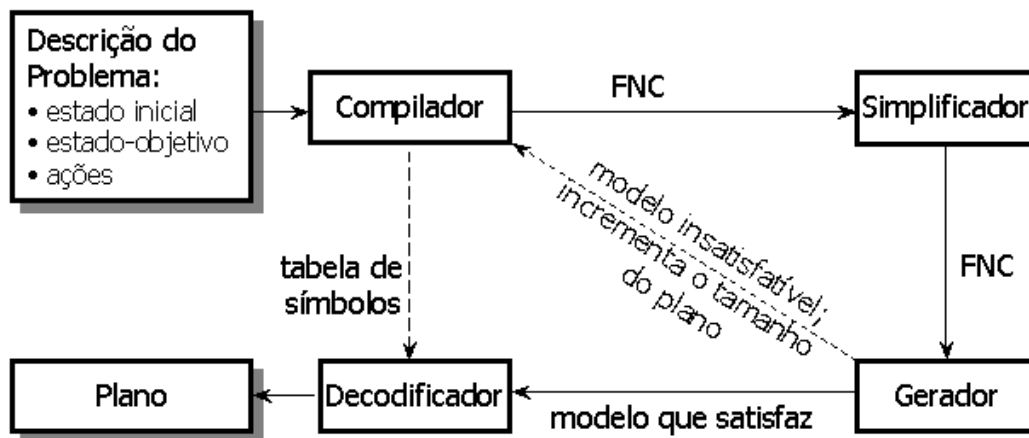



FIGURA 3.3 - Estrutura da Geração de Planos por Satisfatibilidade

Um Compilador recebe a descrição do problema e gera uma Fórmula Normal Conjuntiva (FNC) proposicional equivalente a um plano de tamanho ‘n’. A FNC é uma conjunção de cláusulas booleanas no formato:

```
((estado1 OR estado2) AND (estado3)) ...
```

A FNC gerada é então enviada ao Simplificador. Cabe a ele reduzir o tamanho da fórmula através de técnicas de otimização em tempo polinomial: eliminação de literais e propagação de cláusula única.

O Gerador tenta então encontrar um conjunto de atribuições para os elementos da FNC que resulte numa expressão verdadeira. Caso nenhum conjunto de atribuições torne a expressão verdadeira, o processamento retorna ao Compilador, que aumenta o tamanho estimado do plano e gera uma nova fórmula.

O processo segue até que o Gerador seja capaz de encontrar os valores para tornar a FNC verdadeira. Quando isso ocorre, o Decodificador traduz a fórmula em um plano aplicável ao estado inicial do problema e o encaminha para execução.

3.2.4 Planejamento Baseado em Grafos

Os planejadores baseados em STRIPS constituíram o tipo de sistema de planejamento mais comum até o começo da década de 1990. Mas na primeira edição do IPC em 1998, dos cinco planejadores concorrentes, três adotavam um novo tipo de planejamento, apresentado três anos antes: o planejamento baseado em grafos. Este tipo de planejamento foi proposto por Blum e Furst (1995), e implementado em um sistema batizado de GRAPHPLAN.

O GRAPHPLAN possui duas etapas para a geração de planos: a primeira consiste da geração de um 'grafo de planejamento' com representações das ações e estados. A segunda, de um algoritmo simples de busca exaustiva no grafo gerado. O que o torna especial é justamente a criação do grafo, que reduz consideravelmente o espaço de busca para o algoritmo.

O grafo de planejamento é constituído por níveis, cada um representando um instante no tempo, e possui dois tipos de nós: os que representam estados e os que representam ações. Os nós que representam os estados encontram-se nos níveis pares do grafo. O conjunto de nós de um mesmo nível par representa o conhecimento do mundo em determinado instante.

O primeiro nível do grafo (o nível zero) representa o estado inicial do domínio, e o último nó, o estado-objetivo. Nos níveis ímpares estão representadas as ações cujas pré-condições, em termos de estados, estão presentes no nível imediatamente anterior.

A Figura 3.4 ilustra um grafo de planejamento simplificado para o problema do mundo de blocos, na mesma configuração apresentada na Figura 3.2.

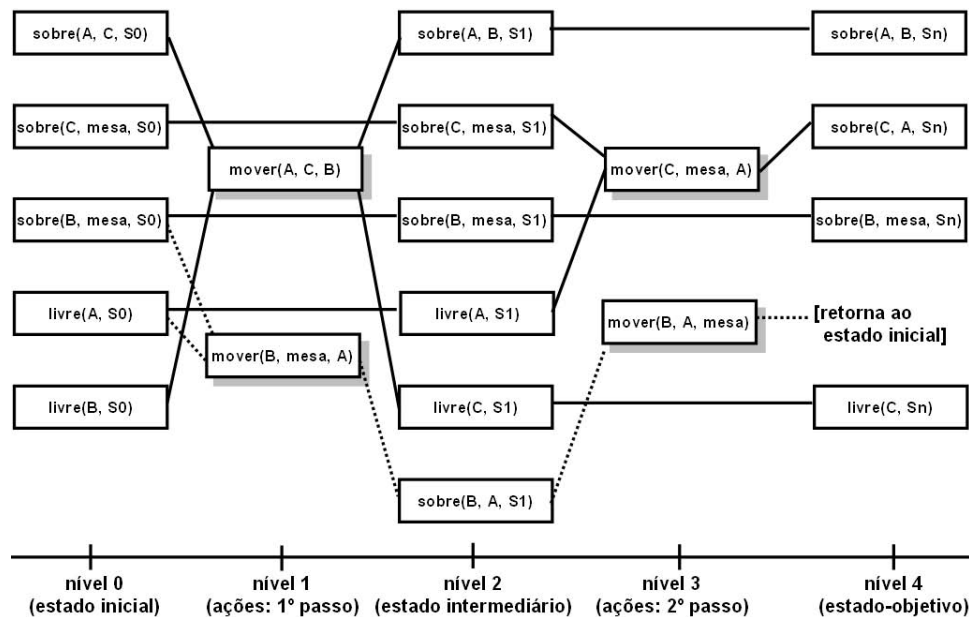



FIGURA 3.4 - Grafo de Planejamento para o Mundo de Blocos

Para gerar este grafo (a primeira etapa da execução), o GRAPHPLAN insere os nós com as literais que descrevem o estado inicial, e em um nível 'n' par, as literais do estado-objetivo. São então verificadas todas as ações cujos operadores de pré-condição (PRE) para o estado inicial são verdadeiros, e estas são inseridas no nível um.

Os operadores ADD e DEL de cada ação são aplicados, e as literais resultantes são colocadas no nível seguinte do grafo (nível dois), junto com as literais não alteradas por ação alguma, que são transferidas para o próximo nível. Novamente são selecionadas ações cujos operadores PRE sejam verdadeiros no nível dois, e inseridas no nível três. Este processo continua até que o grafo atinja as literais do estado-objetivo.

Muitas das ações inseridas no grafo são mutuamente exclusivas (*mutex*), e o controle destas ações é a base da geração de planos por análise de grafos. Duas ações são consideradas *mutex* caso satisfaçam os seguintes critérios:

- Efeitos inconsistentes: o efeito de uma ação é a negação do efeito de outra ação;
- Interferência: uma ação elimina a pré-condição de outra ação;

- Pré-condições inconsistentes: duas ações do nível ‘i’ são geradas a partir de pré-condições que são mutuamente exclusivas no nível ‘i-1’.

O grafo gerado compõe um espaço de busca bastante reduzido com relação ao original. O GRAPHPLAN então dispara um processo de busca exaustiva no grafo para encontrar o melhor plano – que no exemplo da Figura 3.4 é composto por duas ações:

```
mover(A, C, B) -> mover(C, mesa, A)
```

Outros planejadores fizeram uso deste pré-processamento do espaço de busca através da criação do grafo de planejamento, e o combinaram com algoritmos de busca baseados em satisfatibilidade, alcançabilidade e busca heurística, entre outros, com a finalidade de obter melhores resultados.

3.2.5 Combinando Grafos e Satisfatibilidade

Um dos principais planejadores a combinar diferentes abordagens foi o BLACKBOX (Kautz e Selman, 1998), que uniu técnicas de planejamento baseado em grafos e em satisfatibilidade. O BLACKBOX usa o pré-processamento e redução do espaço de estados para a composição de um grafo de planejamento, e aplica a este grafo técnicas de satisfatibilidade na busca pela solução do problema.

O BLACKBOX entrou na competição IPC de 1998, e seu desempenho foi comparável à dos melhores competidores, quase todos baseados no GRAPHPLAN. Entretanto, na edição de 2000 da IPC, seu desempenho foi muito abaixo do esperado. Em 2002 não havia nenhuma versão competindo, e em 2004 uma nova versão do sistema, chamado SATPLAN04, foi considerado o melhor planejador na soma dos resultados.

A variação de desempenho de uma competição para outra foi atribuída às características da solução SAT em relação às características dos problemas apresentados nas diferentes edições da competição. Uma análise detalhada dos motivos desta variação encontra-se descrita em (Kautz, 2006).

3.2.6 Planejamento Baseado em Heurísticas de Busca

Bonet et al. (1997) foram os primeiros a propor o uso de heurísticas de busca na geração de planos. A proposta consiste em realizar uma busca progressiva a partir do estado inicial ao estado-objetivo, utilizando uma função heurística que estima a distância (custo) em passos do estado atual (intermediário) ao estado-objetivo.

A proposta foi implementada no planejador HSP (de *Heuristic Search Planner* – Bonet e Geffner, 1998), que participou da primeira IPC, com bons resultados. A Figura 3.5 mostra um exemplo de planejamento com busca heurística para o problema do mundo dos blocos.

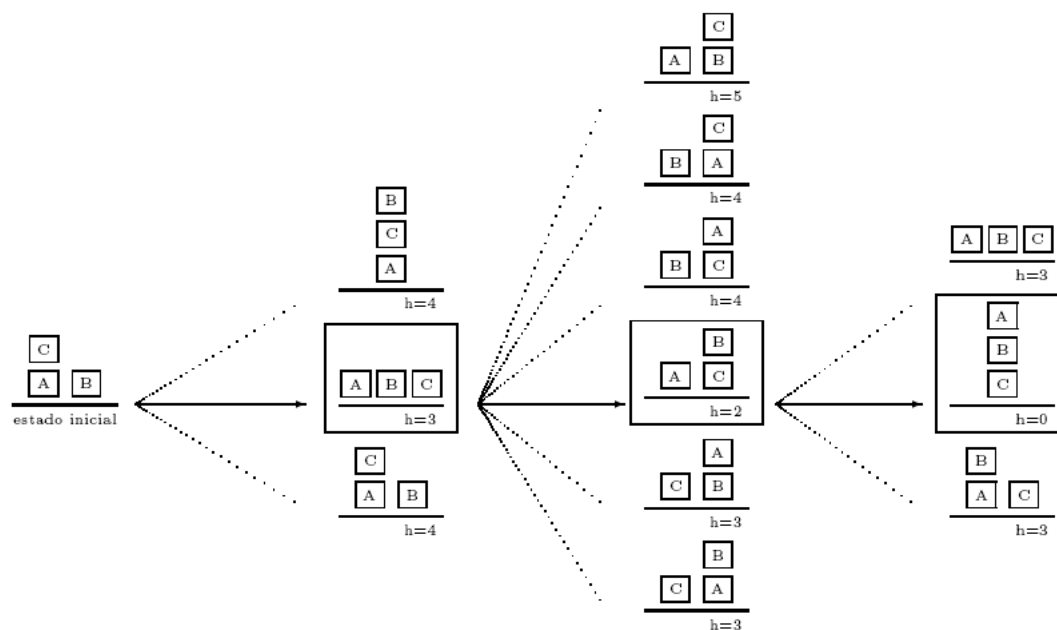



FIGURA 3.5 - Busca Heurística Aplicada ao Problema do Mundo de Blocos

Fonte: Silva (2003, p. 40)

Na Figura 3.5, o estado inicial é definido como `{sobre(C, A), sobre(A, mesa), sobre(B, mesa)}`, enquanto que o estado-objetivo é `{sobre(A, B), sobre(B, C), sobre(C, mesa)}`. Os estados destacados são aqueles que possuem o menor valor definido por uma função heurística h*(s) em cada passo do processo de planejamento, ou seja, aqueles que aparentemente estão mais próximos do objetivo.

Os planejadores aqui apresentados deram origem a diversos novos sistemas, implementando variações ou combinações das abordagens descritas neste Capítulo. Cabe aqui destacar algumas outras áreas de pesquisa em planejamento, como o planejamento probabilístico (Erol et al., 1995), técnicas baseadas em Redes de Petri (Silva et al., 2000) e programação por restrição (Beek e Chen, 1999).

3.3 Formas de Representação do Conhecimento Usadas em Planejamento

Biundo et al. (2003) define Engenharia do Conhecimento em planejamento como o processo que envolve:

- A aquisição, validação e verificação do conhecimento sobre o domínio, bem como a manutenção de modelos de domínios para planejamento;
- A seleção de ferramentas de planejamento apropriadas e sua integração ao modelo do domínio para compor uma aplicação de planejamento.

O 'modelo de domínio' (também chamado aqui apenas de 'modelo') é a base de conhecimento que um agente de planejamento pode utilizar para efetuar decisões racionais sobre o domínio representado.

A forma de representação do conhecimento existente sobre o domínio dita o sucesso de qualquer sistema de planejamento em IA. Classicamente, esta representação (também chamada de 'modelagem') consiste da definição de espaços de estados em que os componentes do domínio possam se encontrar, um conjunto de operadores (ou ações) capazes de alterar estes estados, e um conjunto de regras associadas à execução dos operadores. Os itens seguintes apresentam algumas das principais formas de representação do conhecimento utilizadas em planejamento.

3.3.1 STRIPS: o Paradigma Ainda Vigente

Mais de três décadas depois de sua criação, a forma de representação do conhecimento definida pelo STRIPS e apresentada no item 3.2.1 deste trabalho ainda é o paradigma

vigente na área de planejamento. A esta forma de representação foi dado o mesmo nome do sistema planejador que a implementou pela primeira vez.

A descrição em termos de literais que representam estados, ações, pré-condições e efeitos provou ser capaz de modelar os mais variados domínios e problemas. Diversas extensões surgiram com o passar dos anos, e mesmo novas linguagens, como a ADL e a PDDL – mas estas continuam sendo baseadas nos mesmos conceitos fundamentais.

Entretanto, recentemente a necessidade de se trabalhar com recursos e tempo em planejamento impôs um grande desafio para este tipo de representação, o que vem motivando o estudo de novas formas de modelagem, como o uso de técnicas inspiradas na Orientação a Objetos (item 3.3.4 deste trabalho).

3.3.2 Relaxando as Restrições do STRIPS

A *Action Description Language* (ADL – Pednault, 1989), uma extensão ao STRIPS, foi criada em 1989 com a idéia de tornar a forma de representação do STRIPS menos rígida, permitindo descrever modelos mais complexos. Assim, a ADL adicionou a disjunção em pré-condições, condicionais em efeitos e quantificação universal, tanto nas pré-condições quanto nos efeitos.

Um exemplo do aumento da capacidade de modelagem dos novos recursos da ADL é que passou a ser possível descrever expressões como 'a localização de todas as caixas em um caminhão muda quando a localização do caminhão mudar'[1].

A adoção da ADL por diversos planejadores tornou-a o mais próximo de um padrão para modelagem de domínios que a área de planejamento teve até então. Um dos mais conhecidos planejadores a utilizar a ADL foi o UCPOP (Penberthy e Weld, 1992), que também permitia o planejamento de ordem parcial, onde há instruções sobre como as ações em um plano devem ser ordenadas.

[1] Neste exemplo se usaria o recurso da quantificação para representar 'todas as caixas'.

3.3.3 PDDL: Um Padrão Para a Descrição de Modelos

A ausência de um padrão para a descrição de modelos em planejamento tornava muito difícil comparar a eficácia de sistemas planejadores, quando aplicados a um mesmo problema. Tendo isso em mente, Mcdermott (1998) e o comitê organizador da primeira edição da *International Planning Competition* (IPC) desenvolveram uma linguagem a ser utilizada por todos os sistemas planejadores participantes da competição, de forma a permitir uma comparação empírica entre seus desempenhos. Esta linguagem era baseada na STRIPS e na ADL, e recebeu o nome de *Planning Domain Definition Language* (PDDL). A criação da PDDL melhorou a comunicação de resultados de pesquisa e disparou um aumento no desempenho, expressividade e robustez de sistemas de planejamento.

Desde a versão inicial, algumas extensões para problemas de planejamento mais realistas foram propostos. A PDDL 2.1, usada para a edição de 2002 do IPC, trouxe algum avanço na representação do tempo com ações que possuíam duração – em oposição às ações 'pontuais' existentes até então.

A versão 2.2 implementou literais que permitem a representação de eventos exógenos determinísticos. Eventos exógenos são eventos fora do controle do planejador, mas que afetam o estado do modelo. Em um plano para controlar um carro com o objetivo de levar seus passageiros de um ponto a outro em uma cidade, por exemplo, a abertura e o fechamento dos semáforos encontrados pelo caminho são eventos exógenos, pois estão fora do controle do planejador.

A última versão, PDDL 3.0, permite a descrição de objetivos opcionais, ou seja, objetivos que um planejador deve tentar atingir após ter alcançado todos os objetivos mandatórios. Em geral, nem todos os objetivos opcionais podem ser atingidos, mas um plano que não leve o sistema a estes estados ainda é considerado um plano válido.

A base de conhecimento da PDDL é composta por duas partes, presentes em arquivos distintos: um para a descrição do domínio e outro para as especificações do problema. A descrição do domínio é fixa para todos os problemas, que são compostos por descrições

do estado inicial e dos objetivos a serem atingidos. A Figura 3.6 apresenta um exemplo de domínio em PDDL, representando estados e ações relativos à operação de um satélite.

```
(define (domain satellite)
 (:requirements :strips :equality :typing :fluents :durative-actions)
 (:types satellite direction instrument mode)
 (:predicates
             (on_board ?i - instrument ?s - satellite)
             (supports ?i - instrument ?m - mode)
             (pointing ?s - satellite ?d - direction)
             (power_avail ?s - satellite)
             (power_on ?i - instrument)
             (have_image ?d - direction ?m - mode) )
 (:functions (slew_time ?a ?b - direction))

 (:durative-action turn_to
   :parameters (?s - satellite ?d_new - direction ?d_prev - direction)
   :duration (= ?duration (slew_time ?d_prev ?d_new))
   :condition (and (at start (pointing ?s ?d_prev)) )
   :effect (and  (at end (pointing ?s ?d_new))
                 (at start (not (pointing ?s ?d_prev)))))

 (:durative-action switch_on
   :parameters (?i - instrument ?s - satellite)
   :duration (= ?duration 2)
   :condition (and (over all (on_board ?i ?s))
                       (at start (power_avail ?s)))
   :effect (and (at end (power_on ?i))
                (at start (not (power_avail ?s)))))

  (:durative-action switch_off
   :parameters (?i - instrument ?s - satellite)
   :duration (= ?duration 1)
   :condition (and (over all (on_board ?i ?s))
                       (at start (power_on ?i)))
   :effect (and (at start (not (power_on ?i)))
                (at end (power_avail ?s))) )

  (:durative-action take_image
   :parameters(?s- satellite ?d - direction ?i - instrument ?m - mode)
   :duration (= ?duration 7)
   :condition (and    (over all (on_board ?i ?s))
                      (over all (supports ?i ?m) )
                      (over all (power_on ?i))
                      (over all (pointing ?s ?d))
                      (at end (power_on ?i)))
   :effect (and (at end (have_image ?d ?m)))))
```

FIGURA 3.6 - Exemplo de um Arquivo de Domínio de um Satélite em PDDL

Fonte: adaptada de Cardoso (2006, p. 65)

O domínio mantém toda a semântica do planejamento, com tipos de entidades (classes dos objetos que fazem parte do modelo, definidas na instrução *types* da Figura 3.6), predicados sobre as entidades (ou seja, atributos que definem o estado da entidade, listados em *predicates*), funções utilitárias (*functions*) e ações (*action* para ações pontuais e *durative-action* para ações com duração associada). Cada ação possui descritos seus parâmetros, pré-condições para execução e efeitos sobre o domínio.

É no arquivo de especificações do problema que o domínio é instanciado, através da criação de objetos a partir dos tipos definidos e da definição dos estados inicial e objetivo. A Figura 3.7 apresenta um arquivo de problema para o domínio apresentado na Figura 3.6.

```
(define (problem sat-example)
(:domain satellite)
(:objects
      satellite0 - satellite
      instrument0 - instrument
      image1 - mode
      spectrograph2 - mode
      thermograph0 - mode
      GroundStation2 - direction
      Phenomenon4 - direction)
(:init
      ;; Definição do valor da função
      (= (slew_time Phenomenon4 GroundStation2) 39.73)
      (= (slew_time GroundStation2 Phenomenon4) 40.00)
      ;;Estado inicial do ambiente
      (supports instrument0 thermograph0)
      (on_board instrument0 satellite0)
      (power_avail satellite0)
      (pointing satellite0 GroundStation2))
(:goal (and
      (have_image Phenomenon4 thermograph0))))
```

FIGURA 3.7 - Arquivo de Problema em PDDL para o Domínio do Satélite

Fonte: adaptada de Cardoso (2006, p. 67)

A descrição do problema contém objetos (*objects*), que são instanciações dos tipos definidos no domínio, e as descrições dos estados inicial (*init*) e objetivo (*goal*) do domínio. Para uma análise detalhada deste domínio, consulte (Cardoso, 2006).

3.3.4 OCL: Uma Representação 'Centrada' em Objetos

Proposta por Liu e McCluskey (2000) e implementada em PROLOG, a *Object-Centered Language* (OCL) parte da idéia de se representar modelos de domínios como um conjunto de objetos sujeitos à várias restrições. A idéia de representação baseada em objetos está claramente alinhada com uma abordagem baseada em modelos. Entretanto, a OCL e sua versão hierárquica, OCLh, são totalmente sentenciais: elas ainda descrevem o modelo como um conjunto de predicados, basicamente os agrupando e chamando estes grupos de 'objetos'.

Um modelo em OCL é composto por objetos dinâmicos ou estáticos, agrupados em *sorts*. Isso é o que se chama em Orientação a Objetos (OO) de 'classes', ou na PDDL de *types*. Cada objeto dinâmico existe em um de seus conjuntos de estados (chamados *substates*), caracterizados por predicados. A aplicação de um operador irá mover os objetos de um conjunto de estados para outro. A OCLh adiciona a estes conceitos o relacionamento hierárquico entre *sorts*, permitindo criar objetos a partir da composição de outros, como ocorre em OO. A Figura 3.8 traz um trecho de uma descrição do domínio do mundo de blocos em OCL.

```
sorts(primitive_sorts, [bloco]) .

objects(bloco,
        [bloco1, bloco2, bloco3, bloco4, bloco5, bloco6, bloco7]) .

predicates([
        sobre_bloco(bloco, bloco),
        sobre_mesa(bloco),
        bloco_livre(bloco),
        ]) .

substate_classes(bloco, B, [
      [sobre_bloco(B, B1), bloco_livre(B), ne(B, B1)],
      [sobre_bloco(B, B1), ne(B, B1)],
      [sobre_mesa(B), bloco_livre(B)],
      [sobre_mesa(B)] ]) .
```

FIGURA 3.8 - Exemplo de Descrição de Domínio Para o Mundo de Blocos em OCL

A instrução *sorts* define o tipo 'bloco'. Este tipo é instanciado pela instrução *objects*, onde são criados os objetos 'bloco1' a 'bloco7'. A instrução *predicates* traz os atributos

de um objeto do tipo bloco: ‘sobre_bloco’, ‘sobre_mesa’ e ‘bloco_livre’. Finalmente, a instrução *substate_classes* define explicitamente todos os possíveis sub-estados em que um objeto de determinado tipo pode se encontrar, indicando quais combinações de predicados são legais.

A OCL tem como grande contribuição permitir uma melhor estruturação e maior representatividade do conhecimento sobre o mundo sendo modelado. Mas, por ainda se basear em descrição por predicados, a linguagem é dependente da representação explícita de qualquer mudança nos estados do modelo, exatamente como em STRIPS ou PDDL. De qualquer forma, a OCL representa um grande passo em direção à descrição de domínios de forma totalmente orientada a objetos.

3.3.5 Outras Formas de Representação do Conhecimento

Além das linguagens STRIPS, ADL, PDDL, OCL e suas variantes, existem outras abordagens para a representação do conhecimento em problemas de planejamento. Boa parte delas foi criada tendo aplicações específicas em mente. Um exemplo na área espacial é a *ASPEN Modeling Language* (AML), que será descrita no próximo Capítulo.

3.4 Conceitos de Escalonamento em IA (*Scheduling*)

De acordo com Smith et al. (2000), o conceito comum de escalonamento dentro da área de Inteligência Artificial é de que se trata de um caso especial de planejamento no qual as ações já foram escolhidas, restando apenas selecionar sua ordem e momentos de execução. Segundo eles, esta é uma simplificação infeliz de escalonamento.

Um problema de escalonamento envolve a atribuição de recursos, limitados em quantidade, para tarefas distintas durante determinado período de tempo, de forma a otimizar um ou mais objetivos. Smith et al. (2000) trazem também algumas considerações sobre esta definição:

- Raciocínio sobre tempo e recursos é o que compõe o núcleo do escalonamento. Estes temas receberam, até o final da década passada, pouca atenção da comunidade de planejamento em IA;

- Problemas de escalonamento são quase sempre problemas de otimização. Embora seja fácil encontrar um conjunto de atribuições legais simplesmente aumentando o intervalo entre a execução de tarefas, encontrar um bom conjunto de atribuições é muito mais difícil;

- Problemas de escalonamento também envolvem escolhas. Estas escolhas não estão relacionadas apenas à ordenação de tarefas, mas também à seleção de quais recursos consumir. Para uma determinada tarefa, diversos recursos alternativos podem estar disponíveis, cada qual trazendo diferente custo e/ou duração associadas.

Dado que problemas de escalonamento podem envolver escolhas entre recursos e mesmo entre processos alternativos, o que os distingue dos problemas de planejamento?

A diferença é sutil: enquanto há, nos problemas de planejamento, um grande número de ações a escolher, e complexas interações entre elas, os problemas de escalonamento possuem poucas opções de ação, mas trazem maiores restrições quanto à ordenação das ações. Em um problema de escalonamento é dado um conjunto de tarefas definidas, onde algumas delas podem ser opcionais e algumas podem permitir processos alternativos simples. Em um problema de planejamento, usualmente não se sabe ao menos quantas tarefas (ou ações) são necessárias para se atingir os objetivos.

A pesquisa em escalonamento se originou na área de Pesquisa Operacional (PO) e vem sendo modificada e incorporada à Inteligência Artificial, notadamente na última década. O que distingue a pesquisa em escalonamento da área de PO daquela da área de IA é que em IA o foco tende a ser em formas gerais de representação e em técnicas que cubram diferentes tipos de problemas de escalonamento. Em PO, o foco encontra-se geralmente no desenvolvimento de técnicas para o processamento otimizado de classes específicas de problemas.

Em IA, a abordagem mais comum para a solução de problemas de escalonamento é representá-los como um Problema de Satisfação de Restrições (CSP, do inglês *Constraint Satisfaction Problem*), e aplicar a eles técnicas comuns de satisfação de restrições.

Segundo Russell e Norvig (2004), em um CSP os estados são definidos pelos valores atribuídos a um conjunto de variáveis; para estes valores, são especificados um conjunto de restrições que devem ser respeitadas. Os próximos itens apresentam as principais técnicas utilizadas para a solução de CSPs.

3.5 Técnicas Para a Solução de Problemas de Satisfação de Restrições (CSPs)

Existem diversos algoritmos para a solução de CSPs, mas a maioria deles cai em duas categorias: a busca construtiva ou a busca local. Ambas são descritas brevemente a seguir.

3.5.1 Busca Construtiva

A busca construtiva baseia-se na tentativa de se construir uma solução ao efetuar atribuições a variáveis de forma incremental, verificar as restrições e voltar atrás quando violações nas restrições forem detectadas. A Figura 3.9 traz um algoritmo simples para a resolução de CSPs com busca construtiva.

```
Busca_Construtiva(CSP)
  1. Se qualquer restrição é violada então falhar;
  2. Senão, se todas as variáveis possuem valores atribuídos então retornar(CSP);
  3. Selecione uma variável sem valor atribuído "v";
  4. Escolha um valor "vi" para "v";
  5. Atribua CSP' = propagar(CSP ∪ v = vi);
  6. Busca_Construtiva(CSP');
```

FIGURA 3.9 - Algoritmo para a Solução de CSPs por Busca Construtiva

Fonte: adaptada de Smith et al. (2000, p. 14)

Este procedimento contém um conjunto de técnicas cuja implementação tem enorme impacto no desempenho. São elas:

- Ordenação de variáveis: é a seleção de qual deve ser a próxima variável a selecionar para a atribuição de valores;
- Ordenação de valores: é a seleção da próxima atribuição de valor a testar para uma determinada variável;
- Estratégia de propagação: a propagação é o mecanismo de esboçar conclusões a partir de novas atribuições de valores a variáveis. Estratégias de propagação podem variar de simples regras de verificação de consistência até estratégias poderosas – mas custosas em termos computacionais – de inferência do novo número de restrições entre as variáveis restantes;
- Estratégia de *backtracking*: *backtracking* é o ato de voltar um ou mais passos em um plano em construção para tomar um novo caminho na busca pela solução. Isso geralmente ocorre quando o planejador encontra um ponto a partir do qual não há mais opções de ação a escolher, ou simplesmente quando as últimas ações selecionadas não o levaram a um estado mais próximo do estado-objetivo.

3.5.2 Busca Local

A busca local começa com todas as variáveis do CSP inicializadas com algum valor, sem considerar as restrições impostas ao problema, ou seja, alguns dos valores atribuídos provavelmente estarão violando as restrições. A violação das restrições impostas ao problema também é conhecida como 'conflito'. A idéia básica é modificar gradualmente os valores atribuídos às variáveis envolvidas no conflito, na tentativa de 'reparar' as restrições violadas e levar o plano para mais próximo de uma solução.

Isto é feito gerando-se repetidamente uma 'vizinhança' de novos valores atribuídos, elencando estas atribuições e selecionando a melhor. Cada etapa de criação, avaliação e seleção de um conjunto vizinho de valores é chamada de uma 'mudança'. Como a

eficácia desta abordagem é altamente dependente da atribuição inicial, uma nova atribuição inicial – sempre desconsiderando os possíveis conflitos, para que sejam resolvidos no processo de criação de vizinhanças – é gerada após um determinado número de mudanças terem sido executadas sem alcançar uma solução válida. A cada um destes ciclos, desde a atribuição inicial até atingir o número máximo de mudanças, é dado o nome de 'tentativa'. Após tentativas suficientes sem encontrar uma solução, o processo termina com uma indicação de falha. A Figura 3.10 traz um algoritmo básico de busca local.

```
Busca_Local(CSP)
  1. Repetir por "numero_de_tentativas":
  2.   Gerar uma atribuição inicial, "A", para as variáveis no CSP;
  3.   Repetir por "movimentos":
  4.     Se nenhuma restrição é violada então retornar(A);
  5.     Selecionar uma atribuição na vizinhança, "A'";
  6.     Atribua A = A';
```

FIGURA 3.10 - Algoritmo para a Solução de CSPs por Busca Local

Fonte: adaptada de Smith et al. (2000, p. 17)

Um vizinho é uma atribuição que está próxima da atribuição existente; em geral, é uma atribuição que difere no valor de apenas uma das variáveis (embora existam outras versões desta abordagem). O processo de geração da vizinhança varia em termos de quantos e quais vizinhos são gerados.

Uma solução é selecionar uma variável que está relacionada a uma restrição violada (pode-se dizer que ela 'contribui' para a violação da restrição) e então gerar um conjunto de atribuições que mude apenas o valor da variável selecionada. Uma abordagem mais completa, embora mais custosa, consiste em gerar vizinhos para todas as variáveis que contribuem para a restrição violada.

Uma vez que os novos conjuntos de atribuições (as novas vizinhanças) tenham sido gerados, eles são analisados e elencados. Tipicamente isso é baseado no número de restrições satisfeitas, mas as restrições podem também ser priorizadas de acordo com o

quão importante é satisfazê-las. Seleciona-se então um dos conjuntos de atribuições e repete-se todo o processo, até se eliminar todas as violações.

Algoritmos de busca local podem sofrer de problemas de platôs e de mínimos locais. Platôs são regiões no espaço de busca nas quais nenhuma atribuição aumenta o valor da função objetivo. Mínimos locais são regiões onde todas as atribuições vizinhas são piores que a atual. Nestas regiões, a busca local, pela sua característica de busca nas vizinhanças, é ineficiente. Estes são os principais motivos da geração de um novo conjunto inicial de atribuições a cada tentativa.

A comunidade de IA desenvolveu algumas técnicas para lidar com estes problemas. Um método para escapar de platôs e mínimos locais é a perturbação aleatória. Ao invés de sempre selecionar a melhor atribuição vizinha, um vizinho aleatório é selecionado ocasionalmente. Isso permite eventualmente escapar de platôs e mínimos locais, explorando outras áreas do espaço de busca.

Segundo Smith et al. (2000), estudos empíricos demonstraram que os métodos de busca local freqüentemente resolvem problemas mais rapidamente que a busca construtiva. Por esta razão, estes métodos vêm sendo usados para resolver um grande número de problemas práticos, ou seja, problemas do mundo real. Existem, entretanto, dois pontos fracos a se considerar com relação ao uso da busca local:

- Completeza: com a busca construtiva, se houver uma solução, ela será eventualmente encontrada. Esta garantia não existe para a busca local. Adicionalmente, o procedimento de busca local pode apenas falhar ao encontrar uma solução; ele não pode concluir que tal solução não exista;

- Otimização: é difícil efetuar otimização dentro de um sistema de busca local. Pode-se apenas combinar o critério de otimização com a função de avaliação de novos vizinhos, ou impor restrições mais rigorosas ao problema. Isso, entretanto, tem como custo uma maior dificuldade em encontrar soluções não-ótimas, mas válidas. Deve-se encontrar um meio-termo entre uma solução válida e uma solução ótima para não penalizar excessivamente o processo de busca.

3.6 Unindo Planejamento e Escalonamento em IA

Muitos problemas complexos do mundo real apresentam características tanto de problemas de planejamento, quanto de escalonamento. Para estes, não é suficiente encontrar uma seqüência de ações correta a executar. Estas ações devem ser iniciadas em seus devidos momentos, devem respeitar restrições de ordenação e intervalos de tempo entre outras ações do plano, e devem ter sido escolhidas de forma a consumir recursos compartilhados da melhor forma possível.

Se a década de 1990 foi marcada pelo surgimento de novas abordagens para o planejamento, a tendência da década de 2000 vem sendo a união das áreas de planejamento e escalonamento em IA. O melhor exemplo disso é a mudança, em 2003, do nome da *International Conference on Artificial Intelligence Planning Systems* (AIPS) para *International Conference on Automated Planning and Scheduling* (ICAPS).

Alguns exemplos de sistemas que integram planejamento e escalonamento foram apresentados por Myers e Smith (1999 – para a operação de grandes empresas), Ruml e Fromherz (2004 – controle de manufatura) e Halsey et al. (2004 – planejador e escalonador de uso geral). Há também iniciativas de integração para aplicações na área espacial, que serão apresentadas no próximo Capítulo deste trabalho. A seguir são destacadas algumas das abordagens básicas de integração entre planejamento e escalonamento.

3.6.1 Planejamento e Escalonamento em Cascata

A forma mais simples de integração entre Planejamento e Escalonamento é o modelo iterativo em cascata. Neste esquema, o planejador e o escalonador operam de forma seqüencial: o planejador gera um conjunto completo de ações para atingir um conjunto de objetivos, e então o encaminha para o escalonador, que atribui tempo e recursos às ações.

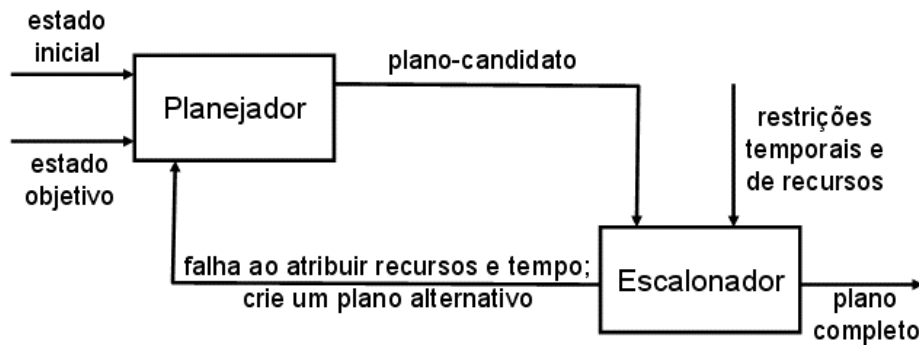



FIGURA 3.11- Planejamento e Escalonamento em Cascata

Logicamente nem todos os planos são escalonáveis e, caso o escalonador não consiga efetuar suas atribuições, ele chama novamente o planejador, para gerar um plano alternativo. Este processo se repete até que um plano seja totalmente escalonável e possa ser encaminhado para execução.

Pode-se facilmente notar que a integração entre planejamento e escalonamento é pequena neste caso, e que o desempenho pode não ser satisfatório. Uma forma de se tentar melhorar a eficiência de um sistema em cascata é aumentar a troca de informações entre o planejador e o escalonador. Ao invés de apenas informar que não conseguiu efetuar as atribuições, o escalonador pode devolver ao planejador dados sobre o que não foi possível atribuir, para que esta informação guie o processo de geração de um novo plano.

3.6.2 Verificação de Viabilidade

A verificação de viabilidade é utilizada para aumentar as chances de que planos gerados sejam viáveis com relação ao escalonamento. Isso envolve o uso das restrições impostas ao escalonador para filtrar, ainda durante o planejamento, as opções de ação a selecionar. Resumidamente, o escalonador é chamado em pontos intermediários do plano para prover estimativas da viabilidade de diferentes alternativas de planejamento.

Esta forma de integração é especialmente adequada para o uso com planejadores hierárquicos, que resolvem o problema em um nível alto de abstração para então partir

para a busca da solução no nível imediatamente mais baixo. Neste caso, o escalonador pode ser acionado ao obter um plano com sucesso em determinado nível, e antes de partir para o nível inferior de abstração. Se o plano for escalonável, o planejador recebe permissão para ir ao nível inferior; caso contrário, ele deverá gerar um plano alternativo no nível atual. É importante notar que, neste caso, o escalonador deve ser capaz de efetuar estimativas de tempo e consumo de recursos em diferentes níveis de abstração.

3.6.3 Planejamento Guiado por Restrições de Escalonamento

Esta é a abordagem onde é maior a integração entre planejamento e escalonamento. Aqui, as decisões do planejador sobre quais ações serão selecionadas para o plano são guiadas não apenas pelas pré-condições das ações, mas também pelas considerações sobre consumo de recursos e tempo provenientes do escalonador.

Torna-se muito mais difícil distinguir o planejador do escalonador, e passa a ser mais importante a utilização de uma representação única do conhecimento do domínio para o planejador e o escalonador.

Com relação a isso, a PDDL apresentou até o momento poucos avanços. Sua versão 2.1 trouxe uma representação temporal ainda limitada, através do conceito de ações com duração. Nelas, pode-se especificar um tempo de duração e efeitos ocorridos ao início ou ao término da ação. Nada há na PDDL com relação ao gerenciamento de recursos.

De fato, poucas linguagens utilizadas em planejamento permitem a descrição de tempo e de consumo de recursos, e isso pode ser, em parte, um efeito adverso do sucesso da IPC e da PDDL: se por um lado elas trouxeram uma padronização na representação do conhecimento e impulsionaram a área de planejamento, por outro elas parecem estar direcionando as pesquisas em planejamento apenas para a busca por algoritmos mais eficientes de solução de problemas. A pesquisa sobre formas alternativas – e potencialmente mais poderosas – de representação do conhecimento para problemas de planejamento segue bastante restrita às extensões da PDDL. Pode-se notar na literatura

uma queda significativa, ocorrida nos últimos anos, em publicações sobre formas de representação não relacionadas à PDDL.

Uma exceção notável à regra vem justamente da área de sistemas planejadores e escalonadores utilizados na área espacial. Como o gerenciamento dos recursos presentes no segmento espacial e o controle temporal da execução de tarefas são partes cruciais dos problemas da área, os criadores destes sistemas tiveram que desenvolver suas próprias representações de recursos e tempo, que serão apresentados no Capítulo seguinte.

CAPÍTULO 4
APLICAÇÕES ESPACIAIS COM PLANEJAMENTO E ESCALONAMENTO EM INTELIGÊNCIA ARTIFICIAL

O segmento espacial, seja ele composto por satélites científicos, observatórios espaciais, sondas ou jipes exploradores em Marte, é controlado através de planos de operação. Isso ocorre de forma igual ou muito similar ao que foi apresentado no Capítulo 2 deste trabalho. Os planos são compostos por seqüências de comandos de baixo nível, ordenados temporalmente de forma a executar determinadas atividades.

Para auxiliar no processo de geração de planos, diversas missões empregam sistemas computacionais, mas estes se limitam a automatizar tarefas simples e repetitivas. Eles pouco ajudam no que se refere às considerações que devem ser feitas sobre as interações entre subsistemas, consumo de recursos, restrições temporais e outros aspectos operacionais. Por isso, a geração dos planos costuma ser um processo basicamente manual e realizado por pessoal altamente qualificado, que deve considerar todas as implicações da execução de cada comando no estado da espaçonave.

Devido a isso, existe o interesse de se aplicar técnicas de planejamento e escalonamento em IA para automatizar a geração de planos. Um dos principais objetivos é reduzir o trabalho dos especialistas, e conseqüentemente os custos da missão.

O Capítulo anterior apresentou as principais técnicas de planejamento e escalonamento em IA[2]. Entretanto, estas técnicas raramente são empregadas de forma 'pura' em aplicações do mundo real. Na área espacial, elas costumam ser combinadas em sistemas híbridos para atender às necessidades de cada aplicação específica.

Este Capítulo apresenta como o planejamento em IA vem sendo estudado e implantado por agências espaciais e instituições a elas vinculadas. São descritos sistemas

[2] A partir de agora, pelo bem da clareza do texto, 'planejamento e escalonamento em IA' será referenciado com freqüência como 'planejamento' ou 'planejamento em IA', uma vez que estas técnicas são tratadas de forma unificada neste trabalho.

planejadores com IA desenvolvidos até hoje, suas características e resultados colhidos de seu uso. Estes sistemas estão categorizados conforme o tipo de automatização que eles proporcionam: a automatização de operações em solo ou o aumento da autonomia da espaçonave. O Capítulo encerra com uma breve apresentação das tendências atuais na área de planejamento aplicado ao espaço.

4.1 Automatização de Operações Versus Aumento da Autonomia

Existem três diferentes níveis de automatização possíveis em uma missão espacial:

1. Geração de Planos de Operação por Operadores Humanos;
2. Geração Automatizada de Planos de Operação;
3. Geração Autônoma de Planos de Operação.

O primeiro nível é onde ainda se encontram a maioria das missões espaciais, nas quais especialistas na missão são os responsáveis pela geração das seqüências de comandos. Neste nível, mesmo que auxiliados por ferramentas computacionais, são os especialistas que tomam as decisões e fazem todas as considerações sobre os efeitos diretos e indiretos da execução de comandos sobre o estado da espaçonave.

No segundo nível, sistemas planejadores com IA são utilizados para gerar os planos, com base no comportamento simulado da espaçonave a partir de modelos descritivos. Neste tipo de automatização, cabe aos especialistas monitorar o processo de geração de planos, corrigindo possíveis falhas ou efetuando melhorias pontuais. Apesar de ainda pequeno, o número de missões que usam este tipo de planejamento vem crescendo nos últimos anos, especialmente na NASA.

O último nível de automatização ainda não é uma realidade para as missões espaciais, mas deve se tornar num futuro próximo. Aqui, a própria espaçonave gera seus planos de operação – ou altera os que lhe haviam sido enviados de solo. Isso permite uma maior capacidade de resposta a eventos externos, otimização no consumo de recursos, e tem o potencial para reduzir drasticamente os custos operacionais da missão. Com sistemas

embarcados de planejamento, as espaçonaves tomam suas decisões – dentro de limites seguros pré-estabelecidos – e as relatam à equipe de operações em solo, geralmente após a execução. Pouquíssimas missões utilizaram planejamento embarcado, e o fizeram em caráter experimental.

O restante deste Capítulo apresenta um histórico da aplicação de planejamento com IA à área espacial, descrevendo planejadores que foram ou estão sendo desenvolvidos para execução tanto em solo quanto em bordo, e o futuro do planejamento na área.

4.2 Automatização de Operações nas Missões Espaciais

4.2.1 Os Primeiros Planejadores com IA para Uso Espacial

A primeira demonstração do uso de planejamento com IA a uma aplicação espacial ocorreu em 1983, quando o planejador DEVISER (Vere, 1983), baseado no NONLIN (Tate, 1977), gerou planos para a operação de uma sonda *Voyager*, da NASA.

O DEVISER foi também o primeiro planejador a trabalhar com representação temporal contínua (cada ação possuía um momento de início e fim), ao invés dos 'momentos' seqüenciais usados por outros sistemas, onde a primeira ação ocorre no momento '1', a segunda no momento '2' e assim por diante.

Por ser apenas uma demonstração, os planos gerados não foram enviados para a sonda. O desempenho do planejador foi considerado insuficiente, e não se deu continuidade, naquele momento, à pesquisa com planejamento em IA.

No final da década de 1980, estudos em planejamento e escalonamento em IA vinham sendo realizados pela ESA dentro de seu programa de pesquisas sobre Sistemas Especialistas. Um dos trabalhos resultantes deste estudo (Fuchs et al., 1988) identificou cinco potenciais aplicações de planejamento e escalonamento em IA ao programa espacial europeu, entre os quais estavam o planejamento de missão para o satélite ERS-1 e o planejamento da montagem, integração e verificação de foguetes lançadores e espaçonaves.

Em resposta a isso, a ESA patrocinou o desenvolvimento de dois planejadores. O primeiro foi o planERS-1 (Fuchs et al., 1990), um protótipo para análise e planejamento de missão para o satélite de sensoriamento remoto ERS-1. O segundo sistema desenvolvido foi o Optimum-AIV (Aarup et al., 1994), utilizado para auxiliar no processo de montagem, integração e verificação de lançadores de satélites Ariane IV. Ambos os planejadores foram baseados na arquitetura O-Plan.

O O-Plan é uma arquitetura aberta de planejamento, desenvolvida em Common Lisp por Currie e Tate (1991). Ele trabalha com decomposição hierárquica, além de aceitar heurísticas para auxiliar o processo de busca. Sua linguagem de descrição é bastante rica, permitindo a representação de restrições de recursos de consumo discreto, contínuo, e renováveis. Além disso, também estavam presentes a representação de tempo e operadores de ordenação (*after*, *before*, etc).

Outra demonstração de planejador com IA em uma aplicação espacial que também foi baseada em O-Plan foi o T-SCHED (Drabble, 1990), desenvolvido para o satélite UOSAT-II, da Universidade de Surrey. Foi gerada uma seqüência de vinte e quatro horas de comandos, que foram enviados ao satélite e executados com sucesso.

4.2.2 Escalonamento de Observações para o Telescópio Espacial Hubble

O telescópio espacial Hubble é um dos maiores e mais complexos sistemas espaciais já desenvolvidos. Ele efetua dezenas de milhares de observações astronômicas a cada ano, propiciando grandes descobertas científicas e imagens espetaculares do Universo.

Qualquer astrônomo, de qualquer nacionalidade, pode solicitar observações a serem efetuadas pelo Hubble. Estas solicitações são enviadas ao *Space Telescope Science Institute* (STScI), órgão responsável pelo gerenciamento das atividades do telescópio. Cabe ao STScI determinar quais das observações solicitadas devem ser efetuadas, aplicando critérios como a importância do alvo a observar, a reputação do astrônomo/instituto solicitante e a viabilidade da observação, entre outros. Uma vez selecionadas as observações, elas devem ser escalonadas da melhor maneira possível.

Cada observação envolve o apontamento do telescópio para o alvo e a manutenção deste apontamento por determinado período. Para efetuar e manter o apontamento em órbita o telescópio faz uso de seus propulsores, cujo propelente é um recurso limitado e de difícil reposição. Logo, é necessário encontrar a melhor seqüência de observações possível, efetuando o mínimo de alterações no apontamento, poupando assim a reserva de propelente e maximizando a vida útil do telescópio.

Mas o consumo de propelente não é o único item a considerar. Cada observação possui uma prioridade atribuída e restrições temporais com rigidez variada. Além disso, o uso de instrumentos diferentes altera constantemente o consumo de memória, energia e alocação dos *links* de comunicação com solo. Fontes de luz intensa como o Sol, a Lua e a própria Terra devem ser evitadas, e restrições ambientais, como a ocorrência de explosões solares e a passagem do telescópio pela Anomalia do Atlântico Sul[3], forçam o desligamento temporário de equipamentos.

Com tamanha complexidade, a geração de planos de observação ótimos é algo quase impossível de ser realizado manualmente. Ciente disso o STScI iniciou, antes mesmo do lançamento do telescópio, o desenvolvimento de dois sistemas escalonadores para automatizar a geração de planos, um para o planejamento de médio/longo prazo e outro para o planejamento de curto prazo.

O *Science Planning Intelligent Knowledge Environment* (SPIKE – Johnston, 1990), é responsável pelo planejamento a médio/longo prazo. Sua tarefa é atribuir observações a ‘blocos de tempo’, alocando primeiramente observações prioritárias, para então preencher os intervalos com observações de menor prioridade.

O resultado é um plano com um ano de duração, mas com observações atribuídas aos ‘blocos de tempo’ (usualmente de uma semana), ou seja, há uma flexibilidade para ajustar cada observação dentro de seu período.

[3] O ponto de maior aproximação do cinturão de Van Allen com a superfície da Terra, onde a intensidade de radiação é maior que em outras regiões da mesma altitude.

O planejador de curto prazo, *Science Planning and Scheduling System* (SPSS – Johnston, 1990), ao contrário do SPIKE, não implementa IA. Ele recebe como entrada o plano gerado pelo SPIKE e atribui tempos específicos para as observações, normalmente para um período de poucas semanas. Ele então traduz este plano detalhado em seqüências de comandos a serem enviados para o Hubble.

O SPIKE, desenvolvido em Common Lisp, representa o problema de escalonamento como um CSP e utiliza uma técnica chamada por Johnston e Miller (1994) de 'reparo estocástico de múltiplos inícios', que é baseado em busca local, auxiliada por heurísticas.

Além do Hubble, o SPIKE foi utilizado em outras missões de observação do espaço, como os satélites *Extreme Ultraviolet Explorer* (EUVE) e o *Chandra X-Ray Observatory*, entre outros.

Outro planejador desenvolvido originalmente para o telescópio espacial foi o *Heuristic Scheduling Testbed System* (HSTS – Muscettola et al., 1995), um projeto conjunto entre o *Jet Propulsion Laboratory* (JPL) e o *Ames Research Center*, ambos da NASA. O HSTS usa decomposição hierárquica e uma forma de busca local batizada como 'reparo iterativo', além de permitir o uso de heurísticas específicas para cada domínio. Ele possui um conjunto rico de restrições temporais e de recursos, o que torna a representação de seus modelos muito expressiva, mas ao custo de uma maior complexidade do planejamento.

O HSTS utiliza variáveis de estado para descrever os componentes do sistema, e as representa como 'linhas do tempo'. Estas linhas do tempo indicam o estado corrente e a evolução no tempo dos valores atribuídos às variáveis.

Ações são posicionadas nas linhas do tempo como *tokens*. Cada *token* possui um momento de início e de fim, sendo que qualquer um destes pode ser colocado como uma 'janela de tempo'. Isso permite, por exemplo, descrever o início de um *token* (uma ação) tanto como 'inicia às 12:03', quanto como 'inicia entre 12:00 e 12:05'.

A busca é parcialmente ordenada, ou seja, a ordem das ações não é totalmente fixa. O planejador explora o espaço de estados sem se comprometer com uma seqüência totalmente ordenada de ações. Isso reduz consideravelmente o tempo de busca (uma vez que as restrições estão relaxadas), e gera planos flexíveis. Entretanto, esta abordagem depende de um executor inteligente, capaz de selecionar, dentro das opções disponíveis no plano, a ordem e momento exato da execução de cada ação.

Apesar de desenvolvido inicialmente para o Hubble, o HSTS foi aplicado às missões EUVE e Cassini, foi usado como base para um planejador embarcado, e deu origem ao ‘motor de planejamento’ EUROPA, que será descrito adiante neste Capítulo.

4.2.3 Planejamento nas Operações dos *Space Shuttles*

Outro projeto espacial extremamente complexo que motivou o desenvolvimento de sistemas planejadores com IA foram os *space shuttles*, também da NASA.

Entre suas missões, cada *shuttle* passa por uma etapa de processamento em solo. Esta etapa envolve a inspeção, manutenção e reparos da espaçonave na preparação para um novo lançamento. Isso implica no planejamento de diversas tarefas, como a alocação de equipamentos, equipes de técnicos e a contratação de empresas prestadoras de serviço.

O planejamento é constantemente alterado, seja pela indisponibilidade de pessoal especializado, por atrasos inesperados, ou pela necessidade de lidar com novos problemas descobertos no *shuttle*. Como as tarefas estão fortemente vinculadas entre si, qualquer alteração na execução de uma tem impacto direto na realização das demais.

Os *space shuttles* são espaçonaves de manutenção extremamente custosa. Qualquer redução no tempo de processamento em solo, mesmo que seja um pequeno percentual do tempo total, representa uma economia de milhões de dólares.

Tendo como objetivo reduzir o tempo de processamento e consequentemente, os custos das missões dos *shuttles*, o *Ames Research Center* da NASA desenvolveu o *Ground*

Processing Scheduling System (GPSS), cujo núcleo, chamado de GERRY (Zweben et al., 1993), implementa escalonamento com técnicas de IA.

Assim como o HSTS, o GERRY representa o problema de escalonamento como um CSP, e faz uso de reparo iterativo baseado em restrições para resolvê-lo. A busca pela solução é guiada com o auxílio de heurísticas específicas para o problema. O domínio é modelado com o uso de variáveis de estado, e a entrada para o processo de busca é um conjunto de tarefas a realizar, cada qual com sua duração, restrições de tempo e de recursos, requisitos de estados para permitir a execução da tarefa e efeitos da tarefa sobre o modelo.

O GPSS recebeu prêmios por sua inovação tecnológica, como o *NASA Space Act Award* e o *Technologies of the Year Award* (1995) da revista *Industry Week*, e seu uso trouxe uma economia de cerca de quatro milhões de dólares/ano para as operações de solo dos *shuttles*.

Mas o uso de sistemas planejadores nos *space shuttles* não se restringiu às operações de manutenção em solo. A missão STS-85, lançada em agosto de 1997, carregou entre suas diversas cargas úteis um conjunto de instrumentos científicos desenvolvidos pela Universidade do Colorado, chamado de *Data Chaser*, com a finalidade principal de efetuar observações solares. O *Data Chaser* possuía um sistema de planejamento automatizado em solo, o *Data Chaser Automated Planning System* (DCAPS – Rabideau et al., 1997), desenvolvido pela Universidade do Colorado em conjunto com o JPL.

No DCAPS, o modelo de domínio é composto basicamente por atividades e recursos. As atividades são utilizadas para modelar tanto os eventos que afetam o estado do sistema, quanto as ações que o *Data Chaser* pode executar. Cada atividade possui uma duração e um conjunto de parâmetros. Além disso, cada atividade pode conter um conjunto de 'sub-atividades', o que permite ao DCAPS realizar planejamento hierárquico. É possível ainda especificar restrições temporais entre as sub-atividades.

Já o conceito de recursos do DCAPS foi utilizado para descrever todos os componentes do sistema modelado. Para isso, foram definidos cinco tipos de recursos:

- Recursos de estado, que representam características (atributos) do sistema que assumem valores discretos. Um subsistema qualquer, como um experimento, pode ser modelado como um recurso de estado, possuindo ao menos dois valores possíveis: 'ligado' e 'desligado';

- Recursos de concorrência, que são utilizados para modelar itens de uso exclusivo no sistema, como um canal de comunicação de acesso único;

- Recursos consumíveis, usados para modelar recursos com quantidade limitada, como propelente e memória;

- Recursos não-consumíveis, que descrevem recursos com quantidade também limitada, mas que é liberada imediatamente após o consumo, como energia;

- E finalmente os recursos simples, que representam dispositivos que só podem ser utilizados por uma atividade por vez.

O DCAPS gera um plano inicial com falhas e aplica a ele técnicas de reparo iterativo, de forma similar ao que ocorre no HSTS e no GERRY. Heurísticas também são utilizadas para auxiliar na decisão de quais alterações devem ser aplicadas ao plano original. Para cada período de operação podem ser gerados mais de um plano, todos atingindo os mesmos objetivos. Estes planos são comparados de acordo com o nível de otimização de cada um. O critério para determinar o nível de otimização num plano é o número de observações científicas realizadas e a quantidade de dados enviados para solo.

Nos primeiros cinco dias da missão, o *Data Chaser* foi operado através de planos gerados manualmente pelos especialistas de missão. Nos sete dias seguintes, a geração dos planos passou para o DCAPS. De acordo com Chien et al. (1998), o uso do DCAPS reduziu em 80% o esforço nas operações relacionadas à geração de planos, em comparação àqueles gerados manualmente, e aumentou o retorno científico em 40%.

4.2.4 Sistemas Planejadores de Uso Geral em Missões Espaciais

Talvez o mais bem sucedido sistema planejador na área espacial, o *Automated Scheduling and Planning Environment* (ASPEN – Fukunaga et al., 1997) é uma evolução do DCAPS que foi criado pelo JPL para ser um ambiente para o desenvolvimento de aplicações de planejamento e escalonamento de propósito geral. Com características herdadas também dos planejadores GERRY e HSTS, e desenvolvido por pessoal que estivera envolvido nestes projetos, pode-se considerar o ASPEN como uma consolidação dos conhecimentos adquiridos pelo JPL nas aplicações de planejamento anteriores da NASA.

O ASPEN é um *framework* (um conjunto reusável de componentes de *software*) que provê as funcionalidades comumente encontradas em sistemas complexos de planejamento e escalonamento. São elas:

- Uma linguagem expressiva de modelagem de restrições, que permite ao usuário definir naturalmente o domínio da aplicação;

- Um sistema de gerenciamento de restrições para a representação da operabilidade da espaçonave e das restrições de recursos, bem como dos requisitos das atividades;

- Um sistema de raciocínio temporal para a expressão e tratamento de restrições temporais;

- Uma interface gráfica para a visualização dos planos.

A linguagem de modelagem criada para o ASPEN é a *ASPEN Modeling Language* (AML). Um modelo descrito em AML possui sete componentes básicos: parâmetros, dependências entre parâmetros, restrições temporais, recursos, variáveis de estado, reservas e atividades.

Um parâmetro é simplesmente uma variável com um domínio restrito e bem-definido. Este domínio pode ser um subconjunto dos inteiros, por exemplo. Outros tipos de

parâmetros aceitos incluem números de ponto flutuante, valores booleanos e *strings*. Uma dependência entre parâmetros é um relacionamento funcional entre dois parâmetros. Um momento de término de uma atividade, por exemplo, é uma função (soma) do momento de início e da duração da atividade.

Uma restrição temporal é o relacionamento entre o momento de início de uma atividade e o momento de término de uma outra qualquer. Pode-se especificar, por exemplo, que uma atividade de preparação de um instrumento deva ser encerrada entre um e cinco segundos antes de uma atividade de uso do instrumento.

Ao contrário do DCAPS, que modelou todos os componentes em um modelo como recursos, no ASPEN recursos representam apenas o perfil de consumo de um recurso físico no tempo. Os tipos de recursos disponíveis são os consumíveis e não-consumíveis, e suas definições são as mesmas do DCAPS. O restante dos componentes é modelado como variáveis de estado, conceito herdado do HSTS, assim como o uso de linhas do tempo para manipulá-las.

Reservas são requisitos para a execução de atividades, com relação a recursos ou variáveis de estado. Por exemplo, uma atividade pode ter uma reserva de dez watts de energia. Isso significa que, durante a atividade, são consumidos estes dez watts. Pode-se especificar se este consumo será calculado no início ou no fim da atividade.

Finalmente, a atividade é a estrutura central no ASPEN. Uma atividade representa uma ação ou um passo em um plano. Ela possui um momento de início, duração e um momento de término. Atividades podem consumir um ou mais recursos e podem conter sub-atividades, o que permite o uso de planejamento hierárquico.

A Figura 4.1 apresenta um pequeno exemplo de modelo descrito em AML, onde estão representadas uma atividade, um recurso, parâmetros, restrições temporais e reservas.

```
Activity prevalve_removal {
        duration = 15
        slot subsystem
        after prevalve_prep with (subsystem == this.subsystem)
        before prevalve_replace with (subsystem == this.subsystem)

        Reservation hydraulic_lift_usage {
                resource = hydraulic_lift;
                usage = 1;
                duration = 5;
        }

        requires state prevalve-purged TRUE
        requires state prevalve-illuminated TRUE
}

Resource hydraulic_lift {
        type non-depletable
        quantity 1
}
```

FIGURA 4.1 - Um Exemplo de Descrição de Modelo em AML

Fonte: Fukunaga et al. (1997, p. 6)

Assim como seus precursores, o ASPEN representa o problema de planejamento/escalonamento como um CSP. Para resolvê-lo, o ASPEN não conta com um algoritmo de busca específico. Ao invés disso, ele suporta uma vasta gama de algoritmos, inclusive algoritmos com *backtracking*, de busca construtiva e de busca local. Este último, considerado pelos autores como o mais importante e chamado por eles de 'algoritmo baseado em reparos', é basicamente o mesmo que foi desenvolvido para o DCAPS, inclusive no uso de heurísticas para auxiliar na tomada de decisões.

O processo de geração do plano possui um tempo-limite (*timeout*). Caso um plano satisfatório tenha sido alcançado antes de atingir este limite, o planejador utiliza o restante do tempo tentando otimizar o plano. Os critérios utilizados para determinar o nível de otimização de um plano são definidos previamente pelo usuário. Nesta etapa, além de tentar melhorar o plano atingido, o ASPEN tenta atingir um conjunto de objetivos opcionais, caso tenham sido especificados no problema. Cada variação viável do primeiro plano gerado é comparada, e ao término do tempo de planejamento a melhor solução encontrada é apresentada.

O ASPEN foi utilizado em diversas missões. Rabideau et al. (1999) destaca o satélite de comunicações navais *UHF Follow-On One* (UFO-1), o satélite de sensoriamento remoto *Earth Observing One* (EO-1), e o satélite universitário *Citizen Explorer One* (CX-1). O ASPEN também possui uma versão embarcada, a ser apresentada ainda neste Capítulo.

Mas o ASPEN não é a única iniciativa da NASA para desenvolver uma ferramenta de planejamento de uso geral. O *Ames Research Center* desenvolveu seu próprio sistema para atingir este objetivo, o *Extensible Universal Remote Operations Planning Architecture* (EUROPA – Frank e Jonsson, 2003).

A idéia por trás do EUROPA foi criar um sistema de planejamento '*plug & play*', que pudesse ser facilmente integrável a outros sistemas e encapsulasse toda a lógica de planejamento e escalonamento, sem deixar de ser extensível para cada missão a que fosse aplicado.

Assim como o ASPEN herdou o legado de outras aplicações de planejamento do JPL, o EUROPA foi desenvolvido sobre os planejadores HSTS e RAX-PS (que será descrito ainda neste Capítulo), ambos do *Ames*. Ele implementa planejamento e escalonamento baseado em restrições, e é integrado às aplicações como um conjunto de bibliotecas em C++.

No EUROPA, um domínio é modelado através de atributos, intervalos, regras de interação e restrições de domínio. Um atributo descreve um componente do domínio, como um subsistema ou equipamento de um satélite. Cada atributo representa uma linha do tempo concorrente, descrevendo sua história através de uma seqüência de estados e atividades. Este conceito é similar ao das linhas do tempo aplicadas às variáveis de estado do HSTS.

Um intervalo descreve um período em que um atributo deve manter determinado valor. As regras de interação são aplicadas às atividades e verificam atributos para definir como subsistemas interagem entre si. As restrições de domínio são impostas à manutenção de valores de atributos em determinados intervalos. Ao contrário do

ASPEN, não há uma forma de representar recursos explicitamente, embora seja possível fazê-lo de forma indireta com o uso de atributos.

O EUROPA vem sendo usado em algumas aplicações de planejamento com sucesso, e foi estendido em uma segunda versão (EUROPA II – Pedersen et al., 2005). Nestas aplicações, ele é comumente chamado de ‘motor de planejamento’ (*planning engine*).

4.2.5 Planejamento em Missões de Exploração de Marte

Nos últimos anos, tem-se observado um aumento nas missões de exploração de Marte. Estas missões têm proporcionado diversas descobertas sobre a geologia e a história do planeta vermelho, seja através de observações realizadas por sondas orbitadoras, ou de dados coletados por jipes exploradores.

Iniciando a recente série de missões à Marte, a missão *Mars Pathfinder* da NASA lançou o jipe *Sojourner* sobre a superfície do planeta em 1997. Apesar do grande feito que foi operar remotamente o jipe através do terreno marciano, um levantamento posterior destacou que o jipe gastava entre 40% e 75% de seu tempo sem realizar atividade alguma, devido à falhas nos planos que lhe eram enviados (Bresina et at., 2002).

Para lidar com este problema, a missão seguinte da NASA para Marte, a *Mars Exploration Rovers* (MER), que pousou em 2004 dois novos jipes (batizados de *Spirit* e *Opportunity*), analisou o uso da tecnologia de planejamento em IA. Embora tenham considerado a tecnologia de planejamento embarcado – que seria mais adequada a este tipo de missão – ainda não madura o suficiente para uma missão deste porte, foi aberta uma oportunidade para o desenvolvimento de planejadores automatizados executados em solo.

Isto levou ao desenvolvimento de um sistema planejador de iniciativa mista chamado *Mixed-initiative Activity Plan GENerator* (MAPGEN – Ai-Chang et al., 2004). Por ‘iniciativa mista’ entenda-se que o MAPGEN interage com os especialistas da missão na criação de planos de operação para os jipes exploradores. A partir de um esboço de

plano, o MAPGEN gera um plano válido e o apresenta ao usuário, que pode efetuar quaisquer modificações que julgar necessárias. Estas modificações podem inserir novos conflitos no plano, que são tratados pelo planejador. O resultado é apresentado novamente ao usuário, e este processo colaborativo segue até que o plano obtido seja considerado satisfatório. O planejamento automatizado é provido pelo ‘motor de planejamento’ EUROPA.

O MAPGEN é utilizado, além do planejamento de operações diárias, para o teste de hipóteses / simulação de cenários, edição de planos e análise do status de recursos através de sua interface gráfica. O uso do MAPGEN na missão MER teve um impacto significativo no tempo de aproveitamento dos jipes. Segundo estimativas levantadas em (Bresina et al., 2005), houve um aumento do retorno científico de 20% a 40%, em comparação com a missão *Mars Pathfinder*.

Para sua próxima missão à Marte, a *Mars Science Laboratory* (MSL), a NASA já está desenvolvendo novos sistemas planejadores, com mais recursos agregados. Um deles, ainda sem nome, utiliza visualização 3D do terreno onde o jipe se encontra (obtida através do processamento de imagens estéreo enviadas por ele) e uma interface gráfica rica para facilitar a geração de planos de operação que envolvam a movimentação e posicionamento de instrumentos sobre alvos a analisar, como rochas, por exemplo (Pedersen et al., 2005).

Neste sistema o planejador utilizado é o *Planning Incrementally for Contingencies* (PICo), que faz uso do motor de planejamento EUROPA II e realiza o que os autores chamam de ‘planejamento de contingência’, onde um plano possui uma linha principal a ser seguida, e um conjunto de ramificações possíveis para lidar com situações de contingência.

Outro sistema em desenvolvimento que tem a MSL como objetivo é o ENSEMBLE (Bresina e Morris, 2006). O ENSEMBLE é a próxima versão do MAPGEN, basicamente atualizando as ferramentas utilizadas e integrando-as melhor. O motor de planejamento utilizado passa a ser o EUROPA II, e o ambiente de desenvolvimento de *software* Eclipse é aplicado para melhorar a integração gráfica entre os módulos.

Além da NASA, a ESA também colabora no esforço científico para aumentar o conhecimento sobre Marte. Sua primeira missão para lá, a sonda orbitadora *Mars Express*, chegou ao planeta em 2004.

Durante os primeiros seis meses de atividade da espaçonave, seus operadores encontraram um grave problema relacionado ao envio dos dados armazenados de carga útil que, em determinados períodos, eram gerados em volume maior do que as janelas de visibilidade com a Terra permitiam transmitir, o que acarretava a perda de dados científicos.

O problema estava sendo tratado manualmente, através da geração de planos de operação que lidavam com cada janela de visibilidade de forma diferente, o que demandava enorme esforço dos especialistas da missão. Para resolver este problema foi contatado o grupo *Planning and Scheduling Team* (PST), do *Italian National Research Council* (Conselho Nacional de Pesquisas Italiano), órgão que mantém relacionamento constante com a ESA.

Este grupo formalizou o problema e propôs o desenvolvimento de uma ferramenta de planejamento com IA para a geração automatizada de planos aptos a tratá-lo. Uma primeira versão desta ferramenta, chamada *Mars EXpress Architecture* (MEXAR – Cesta et al., 2004) começou a ser implantada apenas três meses após ser proposta. O MEXAR foi testado no ambiente operacional da missão e passou a ser utilizado rotineiramente a partir de fevereiro de 2005, em uma nova versão, MEXAR2.

O MEXAR2, assim como o MAPGEN, parte da idéia de planejamento cooperativo. O usuário e o sistema planejador interagem até que um plano satisfatório seja alcançado. A modelagem tem como elementos apenas os recursos e as atividades, e o problema, assim como em outros planejadores, é tratado como um CSP. É utilizado um conjunto de algoritmos executados em série para se atingir a solução. O primeiro algoritmo aplicado é de busca gulosa guiada por heurísticas, que apenas atribui valores iniciais às variáveis do problema, sem tratar as violações de restrições que estejam ocorrendo. A seguir, algoritmos de busca randômica e de busca tabu são utilizados de forma combinada para tratar as violações. É ao término da primeira execução destes

algoritmos que uma primeira versão do plano é apresentada ao usuário. A cada conjunto de alterações sugeridas pelo usuário, os algoritmos de busca randômica e tabu são novamente executados. De acordo com Cesta et al. (2006), o uso do MEXAR2 trouxe um aumento de cerca de 50% na quantidade de dados científicos obtidos da *Mars Express*.

A abordagem de desenvolvimento do MEXAR e MEXAR2 foi focada especificamente para a solução do problema da *Mars Express*, fazendo com que este planejador não seja facilmente aplicável a outras missões. Devido a isso, a equipe do PST está atualmente desenvolvendo um planejador de uso geral, chamado *Open Multi-CSP Planner and Scheduler* (OMPS – Fratini e Cesta, 2006).

Embora não seja voltado especificamente para missões espaciais, os autores pretendem aplicar o OMPS a problemas desta área, e sua experiência prévia com o MEXAR os qualifica para isso. O OMPS é baseado em restrições e traz conceitos de modelagem e algoritmos de busca bastante semelhantes aos do ASPEN.

4.2.6 Estudos em Planejamento Automatizado Realizados no INPE

Existem poucos trabalhos publicados a respeito de técnicas de planejamento em IA na área espacial, fora do eixo NASA–ESA. No INPE, estudos iniciados recentemente buscam o desenvolvimento de ferramentas capazes de automatizar as tarefas de controle em solo de seus satélites.

A arquitetura *Multi-Agent Ground-operation Automation* (MAGA – Biancho et al., 2006) tem como objetivo gerenciar a alocação dos recursos em solo para o rastreio e controle da operação de múltiplos satélites por uma mesma estação terrena de rastreio. Para isso, a arquitetura MAGA combina cinco tipos de agentes:

- Agente de Geração do Problema;
- Agente de Planejamento do Rastreio de Satélites;
- Agente para a Geração do Plano de Operações;

- Agente para a Priorização de Objetivos;
- Agente para a Execução dos Planos em Solo.

Outro trabalho em planejamento com IA no INPE teve como resultado o sistema de Planejamento Inteligente de Planos de Operação de Vôo (PlanIPOV – Cardoso et al., 2006). Este sistema utiliza o planejador temporal LPG-TD, e gera planos de operação para o suporte às atividades de controle de satélites em solo. A linguagem de modelagem utilizada para a descrição do domínio é a PDDL 2.2 – exemplos de arquivos de problema e de domínio do PlanIPOV foram utilizados para os exemplos das Figuras 3.6 e 3.7 do Capítulo anterior.

O PlanIPOV é composto por módulos de geração do problema, configuração, planejamento e visualização. Os módulos de geração do problema e de planejamento serão integrados à arquitetura MAGA, implementando alguns de seus agentes.

4.3 Aumento da Autonomia nas Espaçonaves

Todos os sistemas planejadores descritos até aqui aumentam o nível de automatização na geração de planos de operação. Isso reduz o esforço da geração manual de planos por parte dos especialistas, e tem o potencial para reduzir os custos da missão e aumentar o retorno da mesma.

Entretanto, a automatização em solo é limitada em alguns aspectos importantes da operação de espaçonaves. Em situações de emergência ou no caso da ocorrência de oportunidades científicas (como descrito na Introdução deste trabalho), a equipe em solo pouco pode fazer. Devido aos períodos fora de visibilidade, ao atraso dos *links* de comunicação, e ao próprio tempo de reação do pessoal da missão (para a análise dos dados e tomada de decisão), não é possível dar uma resposta satisfatória a estas ocorrências. Até que um plano possa ser gerado em solo e colocado em execução, os estados do sistema e do ambiente externo a ele provavelmente terão mudado significativamente.

Além disso, a monitoração e o controle de uma espaçonave sempre se baseiam numa visão passada do estado dos subsistemas e do ambiente. Os dados de telemetria contendo os estados podem ter sido coletados até horas antes de sua transmissão, e a quantidade de dados enviados em tempo real é limitada. Torna-se difícil replanejar de forma adequada em resposta a eventos não previstos, tendo à disposição informação defasada sobre a situação do satélite.

Para que seja possível o replanejamento em resposta a ocorrências aleatórias, é necessário manipular informações obtidas em tempo real sobre o estado da espaçonave e de seu ambiente, e colocar o novo plano em execução assim que estiver pronto. A única forma de se fazer isso é transferir o processo de planejamento para o *software* embarcado na espaçonave.

É consenso entre os pesquisadores da área que as espaçonaves serão cada vez mais autônomas. Um levantamento recente sobre os sistemas de execução de comandos em espaçonaves e robôs da NASA (Verma et al., 2005) destaca que "à medida que as missões tornam-se mais complexas, a autonomia torna-se cada vez mais importante para o sucesso destas missões".

Em (Teston et al., 1997), argumenta-se que "é esperado, para um futuro próximo, que atividades de operação de missões evoluam do controle baseado em solo para monitoramento e controle embarcado", e que "no futuro, funções mais complexas serão gradualmente migradas para o computador embarcado. As atividades baseadas em solo de operação de espaçonaves por humanos serão limitadas essencialmente à fase de inicialização da missão e suporte de emergência".

Com o objetivo de aumentar a autonomia de suas futuras espaçonaves, a NASA realizou dois experimentos de planejamento embarcado nos últimos anos. Estes experimentos lhes deram subsídios para iniciar o desenvolvimento de sistemas ainda mais complexos, com metas mais ambiciosas. Os próximos itens apresentam o que foi realizado e o que vem sendo desenvolvido na área de planejamento embarcado.

4.3.1 As Experiências Realizadas com Planejamento Embarcado

A primeira missão a aplicar planejamento embarcado foi a sonda *Deep Space One* (DS-1) da NASA, em 1999 (Jonsson et al., 2000). Isso ocorreu como parte do *Remote Agent Experiment* (RAX), uma demonstração de planejamento com execução em ciclo fechado (*closed-loop*), inferência de estados baseada em modelo e recuperação de falhas.

A sonda DS-1 fez parte do programa *New Millennium* de validação tecnológica para futuras aplicações espaciais da NASA. Sua missão era executar passagens sobre alguns alvos (Marte, um asteróide e um cometa) e efetuar observações durante elas. A principal tecnologia validada pela DS-1 foi seu sistema de propulsão iônica, mas diversas outras foram testadas, como o RAX.

O experimento foi desenvolvido pelo *Ames Research Center*, com colaboração do JPL, e o planejador, chamado *RAX Planner/Scheduler* (RAX-PS), é basicamente uma versão embarcada do HSTS. Assim como o HSTS, o RAX-PS gera planos temporalmente flexíveis, deixando a cargo de um sistema executor inteligente selecionar os momentos exatos de execução de cada tarefa. A Figura 4.2 apresenta a arquitetura do RAX-PS.

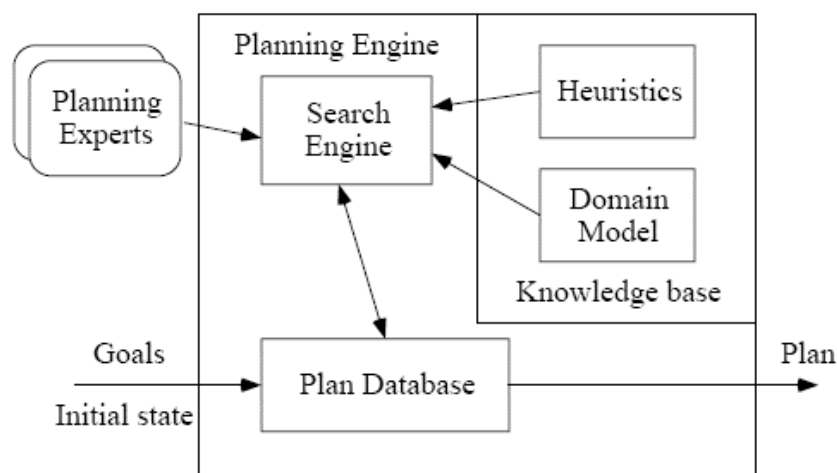



FIGURA 4.2 - A Arquitetura do RAX-PS

Fonte: Jonsson et al.(2000, p. 1)

O modelo de domínio (*domain model*, na Figura 4.2) descreve a dinâmica do sistema para o qual o planejador está sendo aplicado – no caso, da sonda DS-1. Uma solicitação de plano, consistindo de um estado inicial e um conjunto de objetivos a serem atingidos, inicializa o banco de dados do plano (*plan database*), que contém um esboço de plano a ser trabalhado.

O motor de busca (*search engine*) modifica o banco de dados do plano para gerar um plano válido, que é então enviado para um sistema executor inteligente. As heurísticas provêem guiagem para o processo de busca, e os especialistas de planejamento (*planning experts*) permitem a comunicação com sistemas externos, como os de controle de atitude, cujas entradas o planejador deve levar em conta.

O objetivo do RAX-PS foi gerar planos de operação detalhados a partir de comandos de alto nível enviados de solo. Ele foi executado em duas ocasiões no mês de maio de 1999, totalizando cerca de vinte e seis horas de execução. Neste período, o RAX-PS gerou planos para tarefas de mudança de atitude, comunicação, propulsão, observações e navegação, entre outras. O experimento foi considerado um enorme sucesso, e é atualmente referência comum a qualquer trabalho de planejamento e escalonamento, na área espacial ou fora dela. Bernard et al. (1999) traz um relato e uma análise sobre a execução do experimento, inclusive com os problemas ocorridos e as soluções aplicadas.

O segundo sistema planejador embarcado utilizado em uma missão espacial (e até o momento o único outro) foi o *Continuous Activity Scheduling, Planning, Execution and Replanning* (CASPER), uma versão para bordo do planejador ASPEN. Devido às limitações do ambiente embarcado, diversas funcionalidades presentes no ASPEN tiveram que ser removidas do CASPER. Entre elas estão o planejamento hierárquico, *backtracking*, objetivos opcionais e a otimização de planos.

O CASPER seria utilizado no satélite *TechSat-21* da Força Aérea Norte-Americana (Chien et al., 1999), e na constelação de nanosatélites universitários *Three Corner Sat* (3CS – Chien et al., 2001), mas a primeira missão foi cancelada pelos militares e a segunda, perdida durante o lançamento.

Foi apenas com o satélite de sensoriamento remoto *Earth Observing One* (EO-1), que já utilizava o ASPEN em solo, que o CASPER pôde ser executado com sucesso. Assim como a sonda DS-1, o satélite EO-1 faz parte do programa *New Millennium*; enquanto o objetivo da DS-1 era validar novas tecnologias para uso em sondas no espaço profundo, o objetivo do EO-1 era fazer o mesmo para satélites de observação da Terra.

No EO-1 o CASPER faz parte do *Autonomous Sciencecraft Experiment* (ASE – Chien et al., 2003), um experimento que envolve a detecção e a resposta autônomas a eventos de interesse científico, como inundações e erupções vulcânicas. A Figura 4.3 apresenta a arquitetura do ASE.

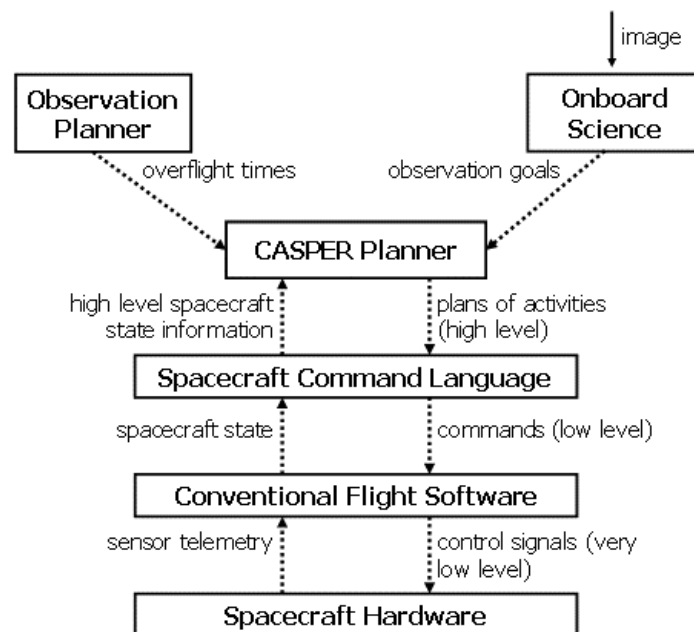



FIGURA 4.3 - A Arquitetura do ASE

Fonte: adaptado de Chien et al.(2003, p. 2)

Para possibilitar a resposta autônoma, o ASE emprega algoritmos de análise de dados de imagens obtidas pelas câmeras do satélite, que identificam a ocorrência dos eventos. Quando da detecção de um evento, uma solicitação de observação é enviada ao CASPER, para que novas imagens sejam feitas da mesma área, nas próximas passagens sobre ela.

O CASPER então consolida os dados de previsão de passagens com os dados do estado do satélite e do plano de operações atual para efetuar o replanejamento. Assim como o RAX-PS, o CASPER também faz uso de um modelo do satélite para inferir seu comportamento durante a geração do plano. Uma vez que o novo plano tenha sido gerado, ele é convertido em comandos para o satélite e colocado em execução. Tudo isso é realizado sem a intervenção da equipe de operações em solo – o CASPER é capaz de gerar uma resposta em algumas dezenas de minutos, tempo suficiente para colocar o plano em execução na próxima passagem sobre o alvo.

Os primeiros testes em vôo foram realizados em outubro de 2003. Testes incrementais foram realizados desde então, até que o sistema foi considerado totalmente operacional em abril de 2005. O uso do CASPER gerou, até o momento, uma economia de cerca de um milhão de dólares na operação do EO-1. Um relato detalhado do processo gradual de implantação pode ser encontrado em Chien et al. (2005).

A Figura 4.4 apresenta uma linha do tempo com todos os sistemas planejadores com Inteligência Artificial aplicados a missões espaciais descritos neste Capítulo.

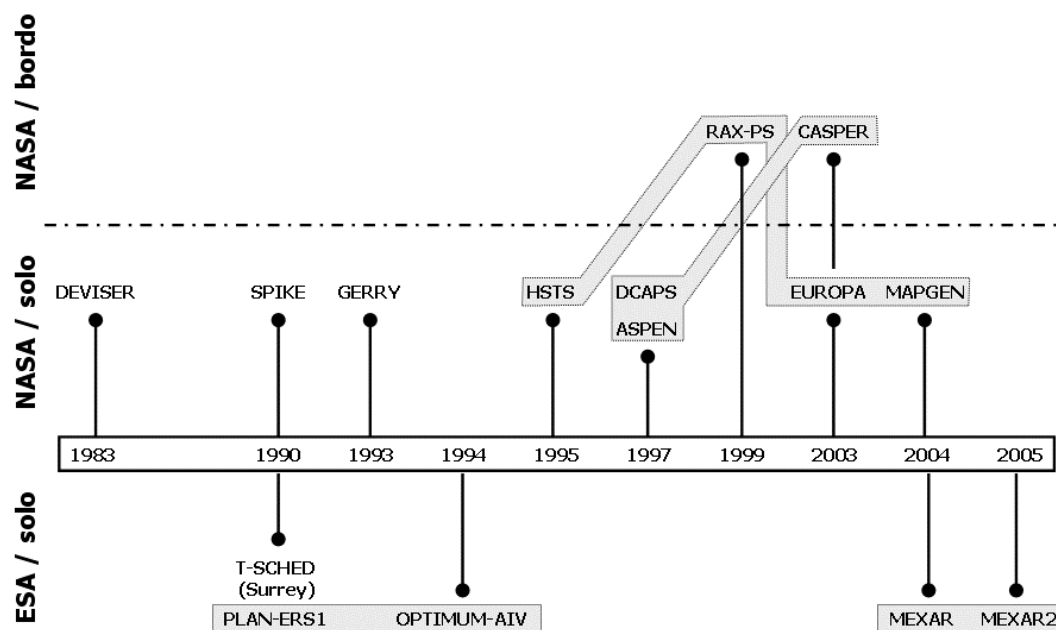



FIGURA 4.4 – Sumário dos Planejadores com IA Aplicados a Missões Espaciais

Os retângulos em cinza destacam as ‘linhagens’ de planejadores. O RAX-PS e o CASPER, que representam o estado da arte em planejamento com IA para a área espacial, são, como se pode notar, as versões embarcadas das duas mais utilizadas linhagens de planejadores em missões espaciais. O ENSEMBLE e o OMPS não estão representados aqui por se tratarem de sistemas ainda em desenvolvimento.

4.3.2 O Futuro do Planejamento Embarcado

O uso de planejamento com IA executado em solo, que foi iniciado com o desenvolvimento de sistemas aplicáveis a uma única missão (como exemplos temos o DEVISER, SPIKE, GERRY e DCAPS) passou a ter como foco, a partir da segunda metade da década de 1990, a busca por *frameworks* genéricos de planejamento em solo, que pudessem ser reaproveitados em diferentes missões. Os produtos disso foram o ASPEN e o EUROPA.

O mesmo vem acontecendo agora com o planejamento embarcado. Após os bem-sucedidos experimentos RAX e ASE, a NASA vem trabalhando em *frameworks* embarcados, que inserem os planejadores já existentes em ‘pacotes de autonomia’. Estes pacotes concentram as funcionalidades necessárias à detecção e resposta a eventos externos, como a ocorrência de fenômenos científicos em órbita ou o encontro com um obstáculo inesperado durante a travessia de um terreno alienígena.

Um destes novos *frameworks* é o *Intelligent Distributed Execution Architecture* (IDEA – Muscettola et al., 2002). O IDEA é uma re-implementação do RAX-PS que emprega o EUROPA como motor de planejamento. Ele tem como objetivo prover um ciclo completo para ‘sentir-planejar-agir’, com resposta em tempo real, na ordem de milissegundos para cada ciclo.

Isso é possibilitado através do conceito de planejamento reativo, onde partes de um plano são liberadas para execução sem que o plano esteja completo. O resultado do trecho do plano executado é lido por sensores e realimentado no processo de planejamento, que gera com isso um novo trecho do plano a executar. Assim, o

planejador reage aos efeitos das partes de seu plano já executadas, adaptando-se aos novos estados lidos. Um plano gerado desta forma não é ótimo, mas permite lidar com situações inesperadas em tempo real.

Outro *framework* que contém planejamento embarcado atualmente em desenvolvimento na NASA (em conjunto com outras instituições) é o *Coupled Layer Architecture for Robotic Autonomy* (CLARAty – Estlin et al., 2005). O CLARAty é um repositório de módulos reutilizáveis de *software* que, juntos, compreendem todas as funcionalidades necessárias para espaçonaves e robôs autônomos. Existem módulos para operação de sensores e aquisição de dados, para o controle motor coordenado, para planejamento, navegação inteligente, entre outros.

Para o planejamento, o CLARAty vinha integrando o CASPER ao sistema executor TDL (Simmons e Apfelbaum, 1998), mas um trabalho recente (Brat et al., 2006) mostra que esta solução foi substituída pelo EUROPA. Pretende-se que o CLARAty se torne um conjunto de bibliotecas de *software* a ser utilizado nos sistemas embarcados de missões exploratórias a serem lançadas em médio prazo – as missões atualmente em fase de desenvolvimento ainda não abriram espaço para o replanejamento embarcado.

Nos últimos dois anos começaram a surgir alguns trabalhos tratando de planejamento embarcado fora da NASA, mostrando que outras agências também passam a se preocupar com o aumento da autonomia de suas futuras missões.

O CNES francês colaborou com um estudo sobre o problema de gerenciamento de constelações de satélites de sensoriamento remoto, no qual um sistema gerenciador global atua em cooperação com planejadores embarcados para efetuar observações de forma coordenada (Damiani et al., 2005). Este, entretanto, é apenas um exercício conceitual.

Na DLR, a agência espacial alemã, um artigo (Axmann e Wickler, 2006) relata o desenvolvimento de um sistema em C++ com planejamento e escalonamento embarcados, a ser aplicado ao satélite *Bispectral InfraRed Detection* (BIRD). Segundo os autores, este sistema, batizado de *Autonomous On-board Mission Planning Software*

(AOMPS), permite a ‘automatização das operações de carga útil’ do satélite. Infelizmente, o artigo é vago demais para permitir qualquer análise mais aprofundada.

O INPE, como apresentado no item 4.2.6 deste Capítulo, já possui uma linha de pesquisa de planejamento com Inteligência Artificial para o suporte às operações de controle de satélite em solo. A proposta deste trabalho de dissertação, a ser detalhada no próximo Capítulo, é dar um passo inicial para os estudos de planejamento a bordo de satélites do INPE visando o aumento de sua autonomia, a partir do desenvolvimento de um protótipo de serviço de replanejamento embarcado.

CAPÍTULO 5

PROPOSTA PARA UM SERVIÇO DE REPLANEJAMENTO EMBARCADO

Este Capítulo apresenta uma proposta para a implementação de um serviço de replanejamento embarcado em satélites do INPE. Primeiramente é mostrado o contexto no qual o serviço de replanejamento embarcado foi inserido, e as características do problema a ser tratado.

A seguir, é dada uma visão geral da arquitetura proposta para o serviço, e as abordagens escolhidas para a forma de representação do conhecimento e para o processo de replanejamento.

O Capítulo encerra descrevendo a forma prevista para a implantação do serviço em uma missão real, através de uma abordagem gradual e baseada na análise dos planos modificados em órbita por parte dos operadores em solo.

5.1 Contexto Definido para o Protótipo do Serviço de Replanejamento

Houve a preocupação, neste trabalho de Dissertação, de se contextualizar o desenvolvimento de um protótipo do serviço de replanejamento embarcado dentro dos projetos do INPE.

Foram identificados dois projetos no Instituto que poderiam ser relacionados a um esforço para o aumento da autonomia do segmento espacial. O primeiro é o projeto de um novo computador de bordo para satélites, escolhido como ambiente de execução e desenvolvimento de *software*. O segundo é o projeto do próximo satélite científico do programa SCE, escolhido para ser utilizado como base para o estudo da modelagem da aplicação. Ambos são apresentados a seguir.

5.1.1 O Projeto do Computador Avançado (COMAV)

O INPE possui diversos projetos de desenvolvimento científico e tecnológico, geridos por diferentes grupos de trabalho espalhados pelo Instituto. Entre estes projetos encontra-se o do Computador Avançado (COMAV), que vem sendo desenvolvido pelo Grupo de Supervisão de Bordo (SUBORD) da Divisão de Eletrônica Aeroespacial (DEA).

O grupo SUBORD participou do projeto e desenvolvimento de todos os computadores embarcados nos satélites do INPE, desde o SCD-1. Sendo um projeto de desenvolvimento tecnológico, o COMAV (Alonso et al., 2001) tem entre seus objetivos a utilização de novas metodologias, técnicas e ferramentas a serem posteriormente aplicadas aos computadores de bordo das futuras missões espaciais do INPE.

A proposta do COMAV, seguindo uma tendência mundial, é unificar as tarefas de supervisão de bordo e o controle de atitude e órbita do satélite, que são usualmente realizadas por computadores distintos. Além disso, prevendo seu uso em missões científicas, o COMAV também irá efetuar o controle e comunicação com experimentos.

O processador utilizado no COMAV é o ERC32 (RISC de 32 bits e arquitetura SPARC V7, que pode trabalhar a até 25MHz), desenvolvido pela ESA especificamente para aplicações espaciais. O uso do ERC32 representa um ganho significativo em termos de poder computacional para os sistemas embarcados em satélites do INPE. Como critério de comparação pode-se destacar que o computador central do CBERS, o mais complexo entre os satélites do INPE, é baseado em um processador CISC 80c86 de 16 bits, que atinge no máximo 4MHz.

Tais velocidades podem surpreender aqueles acostumados aos GHz presentes nos atuais computadores do tipo IBM-PC ou similares. Esta defasagem de desempenho deve-se ao fato de que processadores para aplicações espaciais passam por um rigoroso processo de qualificação, no qual seu próprio projeto está sujeito a modificações, para permitir um menor consumo de energia e maior resistência às altas taxas de radiação do ambiente espacial, entre outros. Este processo pode levar anos para ser concluído, de forma que

os processadores utilizados em missões espaciais nunca são os modelos mais recentes disponíveis no mercado.

Com relação ao *software*, o COMAV irá utilizar pela primeira vez nos satélites brasileiros um sistema operacional de tempo real, gratuito e de código-fonte aberto, o *Real-Time Executive for Multiprocessor Systems* (RTEMS). Os satélites anteriores fizeram uso de sistemas operacionais dedicados, desenvolvidos pelo INPE especificamente para cada missão.

O RTEMS foi criado pela *Online Applications Research Corporation* (OAR) sob contrato com o Departamento de Defesa Norte-Americano para ser um *kernel* de tempo real multitarefa para sistemas embarcados em mísseis. O RTEMS foi posteriormente utilizado em outras aplicações militares e, em 1994, foi aberto para uso em aplicações civis. O RTEMS é atualmente um sistema de código-fonte aberto, de uso livre e que pode ser distribuído e alterado sob os termos da *GNU General Public License* (GPL).

Entre 1995 e 1997 a ESA patrocinou a adaptação do *kernel* do RTEMS para o ERC32 como parte de sua estratégia para a obtenção de independência tecnológica em sistemas espaciais. Atualmente, a versão para ERC32 do RTEMS é homologada para uso em aplicações espaciais.

Por ser um projeto de desenvolvimento tecnológico que propicia maior poder computacional, e por se colocar como um possível computador de bordo para as futuras missões espaciais do INPE, o COMAV apresenta-se como uma oportunidade para que sejam iniciados estudos com planejamento embarcado. Assim, foi definido que o protótipo do serviço de replanejamento embarcado deve ser desenvolvido como um *software* aplicativo do Computador Avançado.

5.1.2 O Satélite Científico EQUARS como Aplicação-Alvo

De acordo com o Plano Nacional de Atividades Espaciais (AEB, 2005) os próximos satélites científicos do programa SCE serão o *Equatorial Atmosphere Research Satellite* (EQUARS) e o Monitor e Imageador de Raios-X (MIRAX). Por ser o satélite em

estágio mais adiantado de projeto entre os dois, o EQUARS foi selecionado como a aplicação-alvo para o estudo da modelagem e análise de cenários do serviço de replanejamento embarcado.

O EQUARS (Figura 5.1) é um projeto de parceria com instituições de diversos países. Os experimentos a serem embarcados serão provenientes de instituições do Brasil, Estados Unidos, Canadá e Japão. O objetivo dos experimentos é realizar o monitoramento global da atmosfera na região equatorial, enfatizando processos dinâmicos, fotoquímicos e mecanismos de transporte de energia entre a baixa, média e alta atmosfera e a ionosfera.

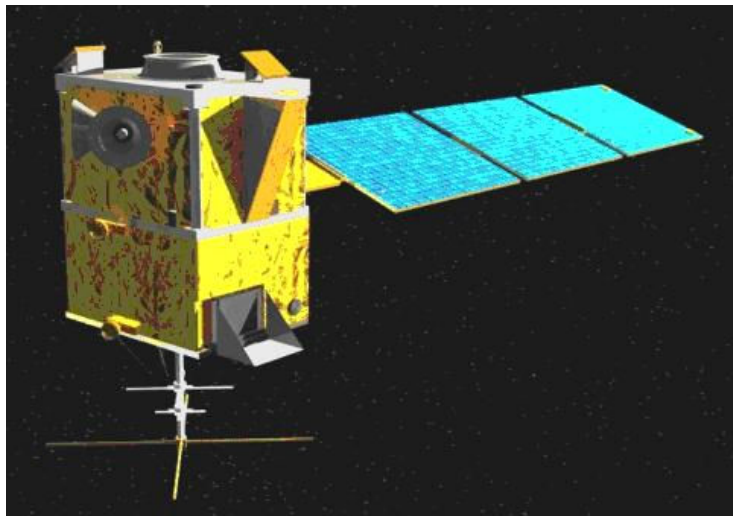



FIGURA 5.1 - O Satélite Científico EQUARS

O conjunto de experimentos científicos previsto para o satélite EQUARS é o seguinte:

- IONEX, um sensor de plasma para a medição de densidade de plasma e temperatura eletrônica;
- GROM, um instrumento baseado na recepção de sinais da constelação GPS para a medição de umidade, temperatura e conteúdo total eletrônico;

- CERTO, um transmissor de *Beacon* que irá fazer observações de irregularidades na ionosfera, conteúdo eletrônico e cintilações;

- TIP, um imageador de luminescência para relâmpagos e *sprites*;

- MLTM, um imageador de temperatura mesosférica.

Os dados a respeito do satélite e de seus experimentos foram obtidos do *website* da missão (INPE, 2005), e das apresentações do Primeiro *Workshop* do EQUARS, realizado em novembro de 2001, também disponíveis no *website*.

5.2 Características do Problema Escolhido

Nas considerações finais sobre o *Workshop on Machine Learning Technologies for Autonomous Space Applications*, realizado durante a vigésima edição do *International Conference on Machine Learning*, Pavlovic et al. (2003) descreve como consenso entre os participantes do *workshop* que "aplicações iniciais [com IA embarcada] devem ser focadas no 'valor agregado' – ou seja, devem permitir que a missão faça mais do que poderia fazer de outra maneira, sem ameaça ao sucesso da missão, ao retorno científico ou à segurança da espaçonave".

O problema escolhido neste trabalho de dissertação, que é a realocação de recursos para experimentos científicos em bordo, segue esta definição. A realocação, caso seja bem-sucedida (quando de uma solicitação por mais recursos), agrega valor à missão, pois permite uma melhor observação dos fenômenos. Caso a realocação não seja possível, os experimentos seguem operando conforme o originalmente programado, sem qualquer tipo de perda. Além disso, o serviço de replanejamento embarcado não é vital à operação do satélite; se ele for retirado ou desabilitado a qualquer momento, nada acontecerá ao restante do sistema.

5.3 A Arquitetura do Serviço de Replanejamento

A partir do estudo das arquiteturas dos sistemas planejadores descritos no Capítulo 4, em especial dos sistemas de planejamento embarcado RAX-PS e CASPER (Figuras 4.2 e 4.3), foram identificados os seguintes componentes básicos, necessários ao serviço de replanejamento:

- Um modelo do sistema, com a descrição das características e comportamento;
- Um módulo para a geração do problema de planejamento;
- Um módulo que implemente o processo de planejamento.

A identificação destes módulos e a definição de suas interações deram origem à arquitetura para o serviço de replanejamento embarcado, que foi batizado de *Resources Allocation Service for Scientific Opportunities* (RASSO). Esta arquitetura é apresentada na Figura 5.2, e descrita brevemente a seguir. Na Figura, o bloco destacado em amarelo corresponde ao serviço aqui proposto. As setas numeradas indicam o fluxo de dados entre os módulos (em laranja) durante o processo de planejamento.

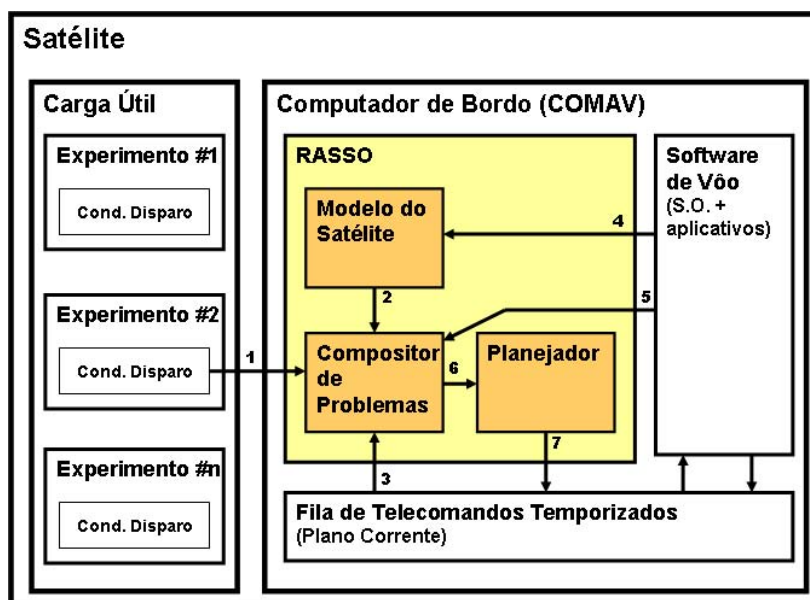



FIGURA 5.2 - A Arquitetura do RASSO

Cada experimento habilitado a solicitar mais recursos do computador de bordo deve ter implementada ao menos uma Condição de Disparo. As Condições de Disparo indicam a detecção de um fenômeno científico, e disparam uma solicitação por mais recursos para o RASSO (seta 1 da Figura 5.2, à esquerda). A forma de implementação e o intervalo de verificação das condições ficam a cargo de cada experimento[4].

Ao receber a solicitação, o módulo Compositor de Problemas do RASSO gera um problema bem-definido na forma de um esboço de plano, com conflitos. Este esboço é criado ao se consolidar informações de várias fontes no *software* embarcado (setas 2 a 5). Os conflitos são inseridos devido à alocação dos recursos solicitados pelo experimento. O esboço de plano é então encaminhado ao módulo Planejador (seta 6).

Cabe ao Planejador trabalhar para a solução dos conflitos, respeitando uma série de restrições que lhe foram impostas e buscando os objetivos que lhe foram passados. Entre as restrições impostas estão as quantidades mínima e máxima de recursos disponíveis, estados de experimentos que devem ser mantidos em certos momentos, e outros. Em suma, os objetivos do planejador serão:

1. Realocar temporariamente os recursos solicitados pelo experimento;

2. Devolver o sistema ao estado normal após a execução do plano alterado em bordo;

3. Fazer isso respeitando todas as restrições de estados e recursos impostas.

Ao obter um novo plano de operações que consiga cumprir todos estes requisitos, o RASSO o encaminha para a fila de telecomandos temporizados (seta 7), substituindo o plano anterior e tornando-o o novo plano de operações corrente do satélite.

[4] Deve-se lembrar que o programa SCE do INPE utiliza o conceito de ‘experimentos inteligentes’, onde cada experimento possui *software* e processador próprios.

5.3.1 A Abordagem para a Representação do Conhecimento

Atualmente as decisões tomadas pelos satélites do INPE são geradas com base em regras colocadas como instruções lógicas. Estas instruções geralmente se apresentam na forma de cláusulas condicionais simples, tais como: 'se a mensagem A chegar, envie a mensagem B', ou 'se a temperatura for menor que 10ºC, ligue o subsistema de aquecimento'. Isso tem a desvantagem de que cada nova situação a ser tratada demanda a inserção de uma nova regra a ser verificada.

Já ferramentas que usam modelos representam sistemas num nível maior de abstração, quando comparados com regras simples. Isso permite a manipulação das informações contidas no modelo, e resulta na capacidade de inferir o comportamento do sistema em resposta a novas situações que lhes forem apresentadas.

A forma de representação do conhecimento utilizada para descrever um modelo pode definir o sucesso ou o fracasso de um sistema planejador. Entre as abordagens para a representação do conhecimento descritas nos Capítulos 3 e 4, a grande maioria utiliza a representação baseada em predicados herdada do STRIPS. Mesmo a PDDL e a OCL, linguagens modernas de planejamento em IA, são fundamentadas nos mesmos conceitos básicos. Estas formas de representação possuem algumas limitações que dificultam seu uso em domínios complexos como o espacial, dentre as quais se destacam:

- A carência no que se refere à representação de recursos e de seu consumo;
- A forma limitada de descrição de pré-condições e efeitos – não há estruturas de *loops* ou seleções de casos, e nem todas as linguagens baseadas em predicados possuem condicionais (*ifs*);
- O fato de que modelos descritos nestas linguagens devem ser interpretados por um analisador (*parser*) e convertidos em estruturas de dados, antes de serem utilizados pelo planejador.

No caso de um planejador embarcado, as opções para lidar com a terceira limitação acima seriam converter o modelo antes de compilar a aplicação, ou adicionar um *parser*

ao *software* do satélite. Tendo em vista que um dos objetivos deste trabalho é buscar a integração entre o planejamento embarcado e o restante do *software* do satélite, qualquer solução que traga mais trabalho ao desenvolvedor, como uma conversão do modelo antes da compilação do *software*, deve ser evitada.

Torna-se desejável então o uso da própria linguagem de programação para descrever o modelo do domínio. Alguns dos motivos para isso são listados abaixo:

- Qualquer linguagem de programação de alto nível já possui recursos avançados de condicionais, seleção de casos e *loops*, que faltam às linguagens de descrição de domínios para planejamento, como a PDDL;

- Outros recursos de linguagens de programação, como chamadas a funções e o uso de ponteiros, podem, se bem utilizados, prover ferramentas de grande valor para o processo de planejamento;

- O uso de estruturas de dados (*structs*) para descrever elementos do domínio leva os modelos de planejamento mais próximos à Orientação a Objetos do que a linguagem OCL, por exemplo, permite. Isso se deve ao fato de que os *structs* foram os precursores dos objetos, tal qual eles são conhecidos em OO.

Entre as formas de representação estudadas, a que mais se assemelha ao que se pretende obter neste trabalho é a AML, utilizada pelo planejador ASPEN. A AML possui uma sintaxe baseada na linguagem de programação C (veja a Figura 4.1), uma vez que o modelo é compilado junto ao planejador, que foi desenvolvido em C++. Na criação de uma aplicação com o ASPEN, um *parser* desenvolvido em Java é responsável por substituir as instruções ao planejador por código C++, transformando um arquivo de modelo em um arquivo-fonte compilável.

Esta etapa de pré-processamento poderia ser evitada caso as instruções ao planejador fossem implementadas por macros da linguagem de programação. Assim, o responsável pela interpretação do modelo passaria a ser o pré-compilador da linguagem de programação.

Foi decidido então pelo uso da própria linguagem de programação para descrever o modelo de planejamento. Macros devem ser utilizadas para implementar as instruções da linguagem de modelagem, e os recursos da linguagem de programação devem ser aproveitados para aumentar a capacidade de representação do domínio.

5.3.2 A Abordagem para o Processo de Replanejamento

A abordagem mais comum para a representação de problemas de planejamento com IA utilizada na área espacial é tratá-los como *Constraint Satisfaction Problems* (CSPs), como foi apresentado no Capítulo 4 deste trabalho. Um esboço de plano é gerado, normalmente a partir do plano de operações corrente da espaçonave. A seguir, as modificações solicitadas para atingir os novos objetivos são aplicadas, inserindo conflitos neste esboço devido à violação das restrições impostas de tempo e recursos.

Aplicam-se algoritmos de busca local (chamados em alguns casos de algoritmos de ‘reparo iterativo’) para corrigir o plano e satisfazer as restrições violadas, uma a uma. Este processo geralmente é incrementado com o uso de heurísticas específicas para o domínio, e de técnicas para escapar de mínimos locais e platôs (definidos no item 3.5.2 deste trabalho) no espaço de busca.

A opção dos desenvolvedores de sistemas planejadores espaciais por esta solução torna-se clara quando se analisa as outras alternativas. Tanto algoritmos baseados em satisfatibilidade quanto aqueles baseados em grafos dependem de uma etapa de pré-processamento, onde o espaço de busca é reduzido. Nesta etapa todas as variáveis do modelo são inicializadas e têm seus valores avaliados em cada momento do tempo, após a aplicação de cada ação possível. Isso demanda uma grande quantidade de memória, que é um recurso bastante limitado em sistemas embarcados.

Além disso, não há implementações de algoritmos de satisfatibilidade ou grafos que gerenciem recursos de forma tão completa quanto os planejadores da área espacial desenvolvidos na última década – embora seus autores argumentem que isso seja possível (Kautz, 2006).

Assim, foi escolhida a representação do problema de realocação de recursos como um CSP, a ser resolvido por um algoritmo de busca local guiado por restrições de escalonamento (item 3.6.3 deste trabalho) e auxiliado por heurísticas específicas para o domínio de satélites. O plano corrente do satélite – enviado rotineiramente a partir de solo – deve ser a entrada para o processo de replanejamento, e a solicitação por mais recursos é responsável por inserir no plano as perturbações que levarão à violação das restrições. Técnicas para a fuga de mínimos locais e platôs no espaço de estados também devem ser implementadas.

5.4 Ganhando a Confiança do Pessoal de Missão

É fato que há uma grande resistência, justificada, quanto ao aumento da autonomia de satélites. O custo e o volume de trabalho em uma missão espacial é em geral considerado muito grande para confiar ao satélite a tomada de mais decisões do que as de rotina, como a correção de atitude e órbita. Esta resistência tende a ser ainda maior quando o aumento de autonomia é propiciado por uma técnica proveniente da área de Inteligência Artificial.

Para lidar com isso e tornar possível uma implementação gradual do serviço em uma missão real, ganhando assim pouco a pouco a confiança da equipe de operações, o RASSO deve trazer duas características:

1. As modificações no plano efetuadas pelo RASSO devem ter escopo reduzido;

2. Deve ser possível à equipe de operações em solo efetuar a análise dos planos gerados, sem que estes sejam colocados em execução.

A primeira característica deve ser provida garantindo que qualquer alteração no plano tenha efeito temporário e que, após o período de observação do fenômeno, o satélite volte ao seu modo de operações normal – ou seja, à forma de operação programada em solo.

A segunda característica deve ser provida através de ‘modos de planejamento’ distintos, que definam o destino dos planos gerados a bordo. O RASSO deve possuir três modos de planejamento: ‘desabilitado’, ‘sugerindo’ e ‘atuando’.

No modo ‘desabilitado’ o serviço não está disponível, e os experimentos sempre têm suas solicitações negadas.

O modo ‘sugerindo’ existe para permitir a análise da confiabilidade do serviço. Neste modo, o serviço recebe as solicitações e sempre envia uma negação aos experimentos, em resposta. Entretanto, ele deve trabalhar num novo plano como se realmente fosse atender à solicitação. O plano resultante não é executado; ao invés disso, ele é disponibilizado para a equipe de operações em solo via telemetria.

Toda vez que uma solicitação tiver sido enviada ao RASSO, quando em modo ‘sugerindo’, uma mensagem de alerta de que há planos armazenados para análise deve ser enviada à equipe de operações, independente se o plano obtido é completo ou não – ou seja, se o Planejador teve ou não sucesso ao gerar um plano que atenda à solicitação.

O plano gerado deve ser enviado em resposta a um telecomando específico, para análise por parte dos operadores da missão em solo. Caberá aos operadores gerar manualmente planos de operação para atender a solicitação do experimento. Os planos dos operadores deverão então ser comparados com aqueles enviados pelo RASSO.

Se for constatado que o RASSO gerou um plano que atenderia à solicitação do experimento, mantendo ainda a operação normal do satélite, isso será computado como um sucesso do serviço. Caso contrário, o algoritmo de planejamento e o modelo do sistema deverão ser revistos e uma nova versão enviada ao satélite. Havendo um número considerado suficiente de sucessos consecutivos, o serviço poderá ser colocado em modo ‘atuando’, passando a estar completamente operacional.

O próximo Capítulo detalha a implementação do protótipo do serviço de replanejamento embarcado aqui proposto.

CAPÍTULO 6

IMPLEMENTAÇÃO DE UM PROTÓTIPO DO SERVIÇO DE REPLANEJAMENTO EMBARCADO

O Capítulo anterior apresentou a proposta para um serviço de replanejamento embarcado, chamado de RASSO, e as abordagens escolhidas para a representação do conhecimento, para o processo de planejamento, e para uma implementação gradual.

Este Capítulo traz a implementação de um protótipo do serviço RASSO. Primeiramente é apresentada a estrutura que foi definida para representar o conhecimento. A seguir, são listadas as principais características da linguagem de representação do conhecimento criada para o protótipo, e utilizada para descrever o modelo do satélite.

Descreve-se então a implementação dos outros módulos do protótipo, o Compositor de Problemas e o Planejador, com ênfase na forma como cada um manipula o modelo.

6.1 Estrutura para a Representação do Conhecimento no RASSO

Conforme descrito no item 3.3 deste trabalho, um ‘modelo de domínio’, ou apenas ‘modelo’, é uma base de conhecimento que um sistema planejador pode utilizar para efetuar decisões racionais sobre o domínio representado. Para o RASSO, foi definido que o modelo do satélite deve ser composto por duas descrições complementares: a descrição estática e a descrição dinâmica.

A descrição estática de um modelo contém a estrutura do modelo, ou seja, os objetos que compõem o satélite (ou a parte dele que se encontra representada no modelo), as classes a que estes objetos pertencem, e os recursos disponíveis para consumo por estes objetos.

A descrição dinâmica de um modelo contém os operadores que modificam o estado do modelo. Estes operadores são de dois tipos: as ações e os comportamentos.

Uma ação corresponde a um ou mais comandos internos do satélite, e descreve como o modelo é afetado por sua execução. Já um comportamento descreve os efeitos sobre o modelo da ocorrência de um evento exógeno (definido no item 3.3.3 deste trabalho). A ocorrência dos eventos exógenos, por sua vez, é representada por janelas de tempo. A Figura 6.1 ilustra a estrutura de uma base de conhecimento no RASSO.

FIGURA 6.1 - Estrutura de um Modelo no RASSO

Os próximos itens descrevem a forma de implementação da linguagem de representação do conhecimento no RASSO, e detalha as descrições estática e dinâmica de um modelo.

6.2 A Linguagem de Representação do Conhecimento

O COMAV está sendo desenvolvido na linguagem C (com o compilador *GNU Cross Compiler* – GCC) e, seguindo a proposta apresentada no Capítulo anterior, é esta a linguagem usada para a modelagem do EQUARS no RASSO.

As instruções ao planejador, que fazem parte do modelo, foram criadas de forma a permitir uma forte integração entre a programação do sistema e o processo de planejamento, tornando a ligação entre o modelo, o planejador e o restante do *software* mais natural. Estas instruções[5] foram implementadas através de macros do C.

[5] Neste trabalho, o termo 'comando' é reservado para os comandos internos do satélite, enquanto que 'instrução' se refere às instruções ao planejador que descrevem as características do modelo.

As macros ocultam a criação e a alimentação das estruturas, vetores, funções e outros elementos utilizados pelo planejador, e fazem com que a descrição do domínio seja mais legível e próxima à linguagens de planejamento, como a AML. A este conjunto de instruções foi dado o nome de RASSO *modeling language*, ou apenas RASSO_ml.

A Figura 6.2 traz exemplos simplificados de instruções ao planejador implementadas na RASSO_ml através de macros. As instruções sublinhadas são utilizadas na descrição do modelo, enquanto que os trechos destacados nas caixas, gerados a partir destas instruções, ficam ocultos do modelador.

FIGURA 6.2 - Exemplos de Instruções ao Planejador Implementadas por Macros

O modelo do domínio é editado junto ao código fonte do *software*. Ele deve estar presente em um arquivo chamado 'model.h', que é vinculado à uma biblioteca RASSO_ml na etapa de pré-compilação do GCC.

6.2.1 A Descrição Estática do Modelo

A parte estática do modelo é composta basicamente por objetos e recursos. Objetos são elementos instanciados a partir de classes, que também são definidas no modelo, e que possuem atributos de qualquer tipo aceito pelo C. O conjunto de valores dos atributos de um objeto em dado instante é chamado de 'estado do objeto'.

Recursos são os elementos consumíveis do modelo. Eles podem ser consumidos pelos objetos ou renovados em função do tempo, sem vínculo a objetos. Há dois tipos de recursos no RASSO: os consumíveis (chamados de *depletable*) e os reserváveis (*reservable*). Ambos os tipos possuem quantidades mínima e máxima definidas em sua criação, e estas quantidades não podem ser excedidas em momento algum.

A diferença entre eles é que, enquanto os recursos consumíveis 'acumulam' seu gasto no tempo até serem esgotados e só são renovados por instruções específicas, os reserváveis são controlados de forma momentânea, sem acúmulo, e são 'repostos' assim que um objeto pára de os consumir.

Durante a operação de um experimento, por exemplo, são consumidos memória e energia. A memória é consumida aos poucos, conforme o experimento coleta e armazena dados – a cada minuto, haverá menos memória disponível, até que os dados sejam transmitidos para solo e a memória seja liberada. Memória é um recurso do tipo consumível.

Já a energia consumida da bateria não é alterada. Se, ao ser ligado, um experimento consumir 10 watts, ele continuará consumindo estes mesmos 10 watts até ser desligado. Assim que isso ocorrer, a bateria volta a ter os 10 watts disponíveis para serem consumidos por outro subsistema. Por isso, energia é um recurso do tipo reservável. Cabe esclarecer aqui que recursos são mensurados em 'unidades', não importando ao RASSO se são watts, quilos ou bytes.

A Figura 6.3 apresenta um trecho de modelo em RASSO_ml com instruções ao planejador para a criação de domínios de valores, uma classe, objetos e recursos. A

criação destes elementos é complementada por uma rotina de inicialização, que se encontra listada ao fim do Apêndice A deste trabalho.

```
create_domain(Exp_Name,
                        certo, grom, ionex, mltm, tip);

create_domain(Mode,
                        full, partial);

create_class(Experiment,
                        Exp_Name        name;
                        bool            on;
                        Mode            mode;
                        int             sample_rate;
                        int             precision;
                        int             priority;
             );

create_object(MLTM, Experiment);
create_object(GROM, Experiment);

create resource(Power);
create_resource(Mass_Memory);
```

FIGURA 6.3 - Criação de Domínios, Classes, Objetos e Recursos em RASSO_ml

6.2.2 A Descrição Dinâmica do Modelo

Os principais elementos da parte dinâmica do modelo são as ações e os comportamentos. Ambos são implementados como funções em C, mas tratados de forma diferente. Este item descreve a estrutura das ações e dos comportamentos, e as instruções ao planejador que cada um pode conter.

6.2.2.1 As Ações

Uma ação (*RASSO_Action*, em RASSO_ml) corresponde a um ou mais comandos internos do satélite, presentes na fila de telecomandos temporizados, e descreve como o modelo é afetado por sua execução. A Figura 6.4 traz um exemplo de uma ação, que liga (*Turn_On*) um experimento passado a ela por parâmetro. A estrutura da ação e suas instruções são detalhadas a seguir.

```
RASSO_Action Turn_On (action_parameters)
{
  parameter(exp);
  when_planning
  {
    // pre-condicoes para a acao
    condition(get_current_state_by_id(Experiment, exp).on == false);

    // instrucao ao planejador
    force_state_if_needed_by_id(Experiment, exp, on, false);

    // efeitos descritos a partir daqui
    set_current_state_by_id(Experiment, exp).on = true;

    switch (get_current_state_by_id(Experiment, exp).name)
    {
      case grom:
      {
        consume_resource_by_id(Experiment, exp, Power, 12);
        consume_resource_by_id(Experiment, exp, Mass_Memory, 100 per_min);
        break;
      }
      case mltm:
      {
        consume_resource_by_id(Experiment, exp, Power, 14);
        consume_resource_by_id(Experiment, exp, Mass_Memory, 264 per_min);
        break;
      }
    }
  }
  when_running
  {
    printf("Turn_On experiment\n");
  }
  action_success;
}
```

FIGURA 6.4 - Uma Ação em RASSO_ml

A ação *Turn_On* recebe como parâmetro um objeto do tipo *Experiment*, representado pela variável 'exp'. Os parâmetros das ações, seus tipos e valores possíveis são informados ao planejador em uma rotina à parte, onde estruturas de controle das ações são inicializadas.

Cada ação possui dois blocos de execução: *when_planning* e *when_running*. O primeiro é chamado durante o processo de planejamento, ao aplicar a ação sobre o modelo de domínio. O segundo bloco deve conter o código que implementa a ação no *software* regular do satélite. Assim, planejamento e execução ficam descritos em um único local,

possibilitando uma maior integração entre o *software* normal e o planejador. Dentro do bloco *when_planning* são descritas as pré-condições necessárias à execução da ação (instruções *condition*) e quais efeitos a ação terá sobre o modelo (*set_current_state_by_id* e *consume_resource_by_id*), caso todas as pré-condições sejam verdadeiras.

6.2.2.2 As Janelas de Tempo e os Comportamentos

As ações permitem descrever os efeitos da execução de comandos de *software* sobre o estado do satélite. Entretanto, nem todas as alterações no estado ocorrem em função de comandos. Alguns dos experimentos do EQUARS têm seu funcionamento vinculado à incidência solar, ou à falta dela. O experimento TIP, por exemplo, faz observações de relâmpagos e *sprites* enquanto se encontra em eclipse (de 'noite', durante uma órbita), o que não é possível quando o satélite está iluminado (de 'dia').

Estes experimentos podem ser ativados e desativados automaticamente por sensores solares, sem que haja comandos 'liga/desliga' agendados para execução. Embora o *software* seja notificado de que o experimento foi ligado/desligado, a informação de quando isso ocorrerá futuramente não se encontra a bordo do satélite, o que é vital para o processo de planejamento. Para lidar com eventos exógenos como os descritos acima, o RASSO faz uso do conceito de janelas de tempo.

Janelas de tempo são períodos nos quais o satélite apresenta determinado comportamento característico, que não pode ser modificado pelo planejador. Por exemplo, durante o período em que o satélite se encontra em contato com uma estação de rastreio em solo, ele transmite os dados armazenados dos experimentos, liberando assim memória de massa. Também consome mais energia, pois seu sistema de comunicação está ativo.

Janelas de tempo podem ser consecutivas ou podem se sobrepor umas às outras (nenhuma restrição quanto a isso é feita pelo RASSO), e não há obrigatoriedade da ocorrência de uma janela de tempo em todas as órbitas. O RASSO possui uma Tabela

de Janelas de Tempo, que armazena os momentos de início e término da ocorrência de cada janela para todas as órbitas nos próximos dias. Estes dados devem ser atualizados regularmente pela equipe de operações em solo. Usando esta informação, a ocorrência das janelas é vinculada aos comportamentos.

Enquanto uma ação deve possuir um ou mais comandos relacionados a ela na fila de telecomandos temporizados, o comportamento (*RASSO_Behavior*) descreve eventos que ocorrem ao início e término de uma janela de tempo, independente do plano de operações corrente. Em outras palavras, os comportamentos descrevem os efeitos da ocorrência de eventos exógenos durante a execução do plano. A Figura 6.5 traz dois exemplos de comportamentos, *Night_Behavior* e *Communicating_Behavior*.

```
RASSO_Behavior Night_Behavior(behavior_parameters)
{
  at_start
  {
    call_action(Turn_On, certo);

    // garante os 5.4 Watts p/ IONEX nos primeiros 12 min. da janela
    guarantee_resource(IONEX, Power, 5.4, 0, 0, 12 * 60);
  }
  at_end
  {
    call_action(Turn_Off, certo);
  }
}
RASSO_Behavior Communicating_Behavior(behavior_parameters)
{
  at_start
  {
    // taxa de transferencia para solo: 230 kbits/s (ou 13.8 Mbits/min)
    generate_resource(Mass_Memory, 13800 per_min);

    // o sistema de comunicacao consome 10W
    consume_resource(Comm_System, Power, 10);
  }
  at_end
  {
    // para de liberar memoria e de consumir energia
    generate_resource(Mass_Memory, 0);
    consume_resource(Comm_System, Power, 0);
  }
}
```

FIGURA 6.5 – Exemplos de Comportamentos em RASSO_ml

Os blocos *at_start* e *at_end* em cada comportamento indicam quais efeitos estão relacionados ao início e ao término da janela de tempo. É possível, inclusive, efetuar chamadas a ações através da instrução *call_action*.

É também possível descrever o consumo de recursos, como ocorre com as ações, e também a sua geração. Na Figura 6.5, a primeira instrução *generate_resource* indica que a memória de massa está sendo liberada à taxa de 13800 unidades por minuto, enquanto o satélite encontra-se na janela de comunicação com solo (ou seja, enquanto está sendo feita a transmissão dos dados de experimentos para solo). Ao término da janela, *generate_resource(Mass_Memory, 0)* indica que a transmissão de dados se encerrou.

Além de descrever os efeitos dos eventos exógenos, comportamentos também são utilizados para impor restrições ao planejador, durante o período de ocorrência das janelas de tempo às quais estão associados. A instrução *guarantee_resource* faz com que seja garantida uma determinada quantidade de recurso para um objeto durante certo período dentro da janela de tempo – no exemplo da Figura 6.5, é imposta a manutenção de pelo menos 5,4 unidades de energia para o experimento IONEX, nos primeiros 12 minutos da janela de tempo *Night*.

6.2.3 Lidando com o Tempo

As ações e comportamentos modificam o estado do satélite à medida que são executados, no decorrer do tempo. Para que o planejador consiga lidar com as mudanças no satélite modelado, ele deve não apenas gerenciar objetos e recursos, mas todos os estados que eles assumem no decorrer do plano, ou seja, todos os seus 'momentos'.

Assim, ao se criar objetos e recursos, na verdade está se criando 'históricos' para eles. Estes históricos são implementados por vetores, que armazenam as modificações sofridas pelos objetos e recursos durante o decorrer do plano. Os históricos são chamados de 'linha do tempo' *(timeline)* para objetos (veja a macro para a instrução

create_object na Figura 6.2) e ‘perfis de consumo’ *(profiles)* para recursos, e são tratadas de forma diferente, como será visto nos próximos itens.

6.2.3.1 Manipulando Objetos no Tempo

Toda vez que uma ação ou comportamento modifica um atributo do objeto, seu novo estado é armazenado no *timeline*, com uma referência à ação responsável pela mudança no estado. A Figura 6.6 ilustra o *timeline* para um objeto ‘MLTM’, que representa um dos experimentos do satélite.

timeline_MLTM[]

	plan_id = 1	plan_id = 2	...	plan_id = n
objeto	name = ionex on = true mode = full sample_rate = 2 precision = 1 priority = 1	name = ionex **on = false** mode = full sample_rate = 2 precision = 1 priority = 1	...	...

a ação nr. 2 no plano desligou MLTM

FIGURA 6.6 - Exemplo de um *timeline* no RASSO

Na Figura 6.6, nota-se que a ação identificada pelo ‘plan_id = 2’ teve como efeito o desligamento do experimento (atributo *on = false*, em negrito).

Por serem armazenados em *timelines*, é necessário que se informe com qual ‘momento’ do objeto se deseja trabalhar, ao obter ou modificar seus atributos. Para isso foram criadas instruções específicas para a manipulação de objetos no tempo. Algumas destas instruções são *get_current_state_by_id* e *set_current_state_by_id* (alguns exemplos se encontram na Figura 6.4). Há também instruções para a obtenção do estado de um objeto no momento inicial do plano, no momento imediatamente anterior ao atual e no estado-objetivo.

6.2.3.2 Manipulando o Consumo e a Geração de Recursos no Tempo

Em RASSO_ml, os recursos são tratados em termos de taxas de consumo/geração. Quando alguma ação ou comportamento modifica a taxa de consumo de um recurso por determinado objeto, a nova taxa é armazenada em um *profile*. A Figura 6.7 apresenta um exemplo do perfil de consumo de energia pelo objeto MLTM.

profile_MLTM_Power[]

plan_id = 5	plan_id = 7	...	plan_id = n
rate = 3	rate = 0	...	...

FIGURA 6.7 - Exemplo de Perfil de Consumo de Recurso

No exemplo da Figura 6.7, a ação de número 5 no plano teve como efeito o consumo de energia pelo objeto MLTM à taxa de 3 unidades por segundo (o segundo é a unidade de tempo básica do RASSO). Esta pode ser, por exemplo, uma ação que ligue o experimento. A ação de número 7, por sua vez, tornou o consumo de energia igual a zero. Esta pode ser uma ação do plano que desligue o experimento. Estruturas similares à da Figura 6.7 armazenam as taxas de geração de recursos, que, diferentemente do consumo, não são vinculadas a objetos (veja a instrução *generate_resource* na Figura 6.5).

Cabe destacar aqui que a modelagem de consumo/geração de recursos em termos de suas taxas, utilizada na RASSO_ml, difere de outras abordagens de planejamento.

O consumo de recursos em problemas de planejamento está sempre vinculado às ações. As ações foram até hoje tratadas de duas formas distintas: como ações pontuais, de efeito imediato sobre estados e sem consumo de recursos, ou como ações com duração definida e consumo de recursos associado. Exemplos deste segundo tipo são as *durative actions* da PDDL e as *activities* da AML. Chamaremos este tipo de ação de ‘atividades’, para distingui-las das ações sem duração.

O ponto fraco do conceito de atividades é que a modelagem do consumo de recursos é discreta (Fukunaga et al., 1997), e reduz as possibilidades de manipulação pelo

planejador. Para explicar a diferença entre as abordagens, tomemos como exemplo uma atividade 'Experimento_Operando', com uma duração de 60 minutos, e cuja execução é responsável pelo consumo de 120 Mb de memória.

Em RASSO_ml, esta mesma atividade seria modelada como duas ações. A primeira, 'Ligar_Experimento', informa ao planejador que o consumo de memória por determinado objeto, a partir daquele instante, é de 2 Mb por minuto. A segunda, que estaria posicionada no plano 60 minutos após a primeira, seria 'Desligar_Experimento'. Ela apenas informa ao planejador que o objeto não consome mais memória.

A diferença básica entre as abordagens, com relação à modelagem do consumo de recursos, é que ao trabalhar com atividades, caso o planejador precise modificar a execução de 'Experimento_Operando' para alocar 20 Mb de memória para outra atividade, suas únicas opções serão deslocar a atividade temporalmente, ou simplesmente removê-la do plano.

Na abordagem da RASSO_ml, onde uma ação não possui duração, mas seus efeitos sim, o planejador poderia manter inalterada a ação 'Ligar_Experimento', e adiantar a execução de 'Desligar_Experimento' em dez minutos, garantindo a memória necessária. Isso maximiza o tempo de operação do experimento, sem deixar de garantir os recursos necessários para outro. Esta abordagem é também mais próxima à forma de operação real do satélite do que as demais.

6.3 O Compositor de Problemas

Cada experimento que puder detectar fenômenos de curta duração deverá possuir pelo menos uma Condição de Disparo em seu *software* que, quando verdadeira, irá enviar uma solicitação por mais recursos ao RASSO. Esta solicitação deve informar:

- Quais recursos o experimento precisa;
- As quantidades necessárias de cada recurso;
- A partir de que momento os recursos devem estar disponíveis;

- Por quanto tempo os recursos alocados devem ser mantidos.

A solicitação por recursos será recebida pelo Compositor de Problemas, que é responsável por reunir informações de diversas fontes, e fornecer ao Planejador um problema a ser resolvido, conforme ilustrado na Figura 6.8.

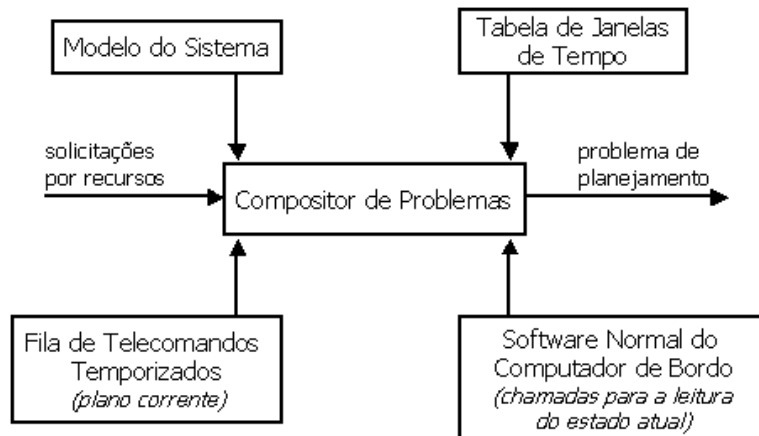



FIGURA 6.8 - Consolidação de Dados para a Composição do Problema

A criação de um problema bem-definido é tão importante quanto o processo de busca pela solução. O problema consiste de um esboço de plano de operações com objetivos a serem atingidos, restrições a serem respeitadas e conflitos a serem resolvidos. Fazem parte do problema também o estado inicial do satélite, os estados objetivo, os telecomandos (as ações, na RASSO_ml) agendados para execução e os eventos exógenos (os comportamentos) que ocorrem durante o período de execução do plano.

6.3.1 Horizontes de Planejamento

A solicitação de recursos enviada pelo experimento carrega em seus parâmetros a informação de dois momentos cruciais para o processo de planejamento: o início e o término do período em que o experimento precisa de mais recursos. Estes 'momentos-chave' são chamados de horizontes de planejamento, e são utilizados para determinar o estado inicial, os estados intermediários e o estado objetivo (Figura 6.9).

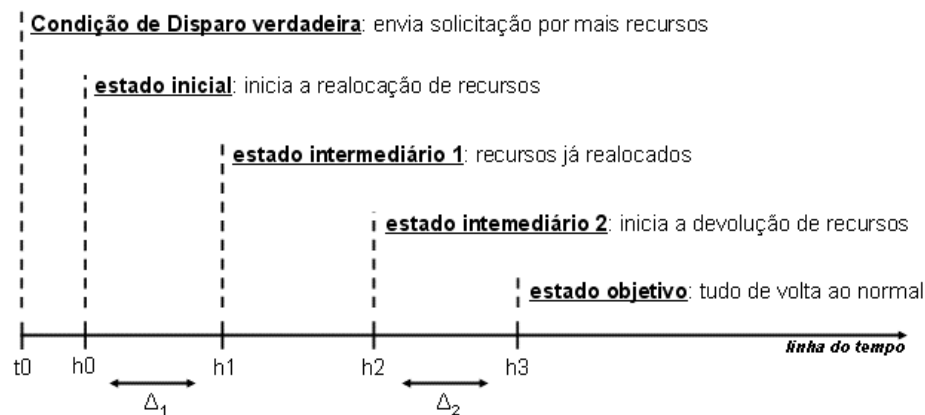



FIGURA 6.9 - Horizontes de Planejamento e Objetivos

O momento de início do período com mais recursos alocados para o experimento solicitante é chamado de 'horizonte h1', ou simplesmente 'h1'. O momento de término do período com mais recursos é 'h2'. Os horizontes h1 e h2 indicam os momentos em que estão os estados-objetivo intermediários do processo de planejamento.

O horizonte h1 indica o momento em que a quantidade solicitada dos recursos já deve estar realocada para o experimento. Já h2 indica até quando estes recursos devem ser mantidos. Em função destes, há mais dois horizontes: o horizonte inicial 'h0' e o horizonte final 'h3'.

O horizonte inicial h0 é definido como sendo h1 – Δ1, onde Δ1 é o tempo necessário para a execução de comandos de realocação de recursos para o experimento solicitante. Se um experimento houver solicitado mais energia a partir de h1, por exemplo, o planejador pode começar a inserir comandos de realocação a partir do momento h0.

De forma similar, h3 é determinado por h2 + Δ2, onde Δ2 é o tempo necessário para devolver os recursos aos experimentos que os cederam. Seguindo ainda o exemplo anterior, se o experimento solicitou que a energia alocada seja mantida para ale até o momento h2, o planejador terá, a partir de h2, até o momento h3 para inserir comandos no plano que realoquem novamente os recursos, e assim colocar o satélite em modo 'normal' de operações.

É preciso definir o que significa ‘modo normal de operações’ aqui: como o RASSO se propõe a alterar temporariamente o modo de operações do satélite, o plano gerado por ele deve garantir que, ao atingir o horizonte h3, todos os objetos modelados estejam no mesmo estado que se encontrariam caso tivesse sido executado o plano original, e que a quantidade de recursos disponível seja pelo menos igual àquela que seria deixada pela execução do plano original.

6.3.2 Compondo um Problema Bem-Definido

A composição do problema passa por três etapas principais: a obtenção do estado inicial do modelo no horizonte h0, a obtenção do plano de operações corrente do satélite e o levantamento dos efeitos de sua execução sobre o modelo entre h0 e h3, e a imposição de restrições e estados-objetivo intermediários e final ao planejador, incluindo os conflitos a serem resolvidos. Estas etapas são detalhadas a seguir:

Obtenção do estado inicial do modelo em h0:

1. O modelo é inicializado;

2. O estado atual dos objetos do modelo é obtido através de chamadas a outros processos aplicativos do *software* de bordo do satélite;

3. A fila de telecomandos temporizados é lida. Todas as ações relacionadas aos comandos na fila são aplicadas ao modelo, conforme suas *tags* temporais, do momento atual até o horizonte h0;

4. A Tabela de Janelas de Tempo é lida. São verificadas as janelas que iniciam ou terminam até h0. Os comportamentos vinculados a estas janelas de tempo são aplicados ao modelo, do momento atual até h0.

Com isso, obtém-se o estado inicial. O Compositor do Problema passa então à obtenção do plano de operações corrente e de seus efeitos sobre o modelo:

5. Repete-se o passo 3, desta vez obtendo e aplicando os comandos entre os horizontes h0 e h3. Todas as alterações nos estados dos objetos e recursos (linhas de tempo e perfis) são armazenadas;

6. Repete-se o passo 4, agora obtendo os comportamentos entre h0 e h3. As instruções *guarantee_resource* (descritas nos comportamentos), não são aplicadas neste momento. Novamente as alterações nas linhas do tempo e perfis são armazenadas;

Tendo o plano de operações corrente do satélite e seus efeitos, aplica-se as restrições ao planejador:

7. O estado do satélite ao término do período de execução do plano (horizonte h3) é marcado como 'objetivo final'. Isso faz com que o planejador respeite este estado e procure ações que levem o satélite a eles, durante o processo de planejamento;

8. É aplicada, ao esboço de plano, a solicitação por mais recursos enviada pelo experimento. Isso é feito impondo como restrição ao planejador que ele mantenha pelo menos a quantidade de recursos solicitada pelo experimento durante o período entre h1 e h2;

9. São aplicadas as instruções *guarantee_resource*, contidas nos comportamentos. Assim como a solicitação por recursos, elas impõem restrições que devem ser respeitadas. A diferença é que a solicitação por recursos informa apenas uma quantidade mínima a ser mantida para o experimento, enquanto *guarantee_resource* pode impor que os recursos garantidos se mantenham entre determinados valores mínimo e máximo;

Ao finalizar as etapas descritas acima, o Compositor de Problemas encaminha ao módulo Planejador o problema na forma de um esboço de plano de operações com objetivos a atingir, conflitos a resolver e restrições a respeitar.

6.4 O Planejador

Conforme apresentado no Capítulo anterior, foi definido que o problema de planejamento deve ser representado como um CSP, e que deve ser aplicada busca local para encontrar a solução. O esboço de plano encaminhado pelo Compositor de Problemas possui associadas a ele estruturas de dados que permitem o seu tratamento como um problema de satisfação de restrições.

Ao receber o esboço de plano, o Planejador entra em um *loop* em busca da solução dos conflitos detectados. Para cada conflito, diversas opções para a solução são testadas e avaliadas, e a melhor entre elas é selecionada. Este processo segue até que não haja mais conflitos no plano. O algoritmo é apresentado em pseudo-código na Figura 6.10.

```
Plan()
{
    Obter_Momentos_Chave_no_Plano();
    numero_conflitos = Obter_Conflitos_no_Plano();

    Enquanto (numero_conflitos > 0):
    {
        Para cada momento-chave no plano:
        {
            Obter_Conflitos_no_Momento(momento);

            Para cada conflito neste momento:
            {
                Obter_Contribuintes();

                Para cada contribuinte deste conflito:
                {
                    conflito_resolvido = Testar_Mudancas(momento, conflito, contribuinte);
                }
            }
        }

        Se (conflito_resolvido = false)
        {
            Inserir_Perturbacao_no_Plano();
        }

        Escolher_Melhor_Mudanca();
        Aplicar_Mudanca_ao_Plano();

        numero_conflitos = Obter_Conflitos_no_Plano();
    }

    Se (modo_planejamento = sugerindo)
        Disponibilizar_Plano_Para_Telemetria();
    Senao (modo = atuando)
        Enviar_Plano_Para_a_Fila_TCs_Temporizados();
}
```

FIGURA 6.10 - O Algoritmo de Planejamento

O algoritmo de planejamento criado procura solucionar o CSP através de busca local guiada por restrições de escalonamento (cuja definição encontra-se no item 3.6.3 deste

trabalho), com a adição de perturbações no plano quando necessário, para escapar de mínimos locais e platôs. As perturbações consistem da inserção de ações que resolvam um conflito, mas cujos pré-requisitos ainda não são verdadeiros no plano. Elas serão melhor explicadas ainda neste Capítulo.

Heurísticas para auxiliar no processo de planejamento não foram inseridas neste protótipo. Os itens seguintes descrevem este algoritmo e seus elementos.

6.4.1 Os Momentos-Chave na Linha do Tempo do Plano

O processo de busca por um plano válido tem início com a identificação de um conjunto de 'momentos-chave' na linha de tempo do plano, e na verificação da ocorrência de conflitos nestes momentos. Os momentos-chave são:

- Os horizontes de planejamento: h0 (início do plano), h1, h2 e h3 (término do plano), onde são impostos estados e quantidades disponíveis de recursos a respeitar;
- Os momentos de execução de ações no esboço de plano recebido do Compositor de Problemas;
- Os momentos de início e término de janelas de tempo que ocorram no período entre h0 e h3;
- Os momentos de início e término das restrições de recursos, impostas pelas instruções *guarantee_resource* e pela própria solicitação do experimento.

Tendo levantado estes momentos, o planejador passa a verificar se existem conflitos em cada um deles, de h0 até h3. Isso é feito aplicando sobre o modelo as ações do esboço de plano e dos comportamentos vinculados às janelas de tempo que ocorram entre h0 e h3, e verificando através de seus efeitos se alguma restrição é violada.

6.4.2 A Identificação dos Conflitos

Mais de um conflito pode ocorrer a cada momento-chave. Os conflitos podem ser de três tipos:

- Estados inconsistentes;
- Violação das restrições de consumo total de recursos;
- Violação das restrições de consumo de recursos por objetos.

Um conflito de estados inconsistentes diz respeito a estados impostos ao planejador nos horizontes h1, h2 e h3. Quando a execução das ações em um plano e dos comportamentos não leva aos estados impostos nestes momentos, há um conflito a resolver. Os conflitos de estados são verificados pela comparação simples entre os estados atingidos pela execução do plano até o momento, com aqueles que são esperados (impostos) para este momento.

Um conflito de violação em uma restrição de consumo total de recurso diz respeito ao uso excessivo do recurso pela soma dos objetos, consumindo além de sua quantidade disponível (máxima).

O último tipo de conflito, o de violação em uma restrição de consumo de recurso por um objeto, ocorre quando algum objeto consome uma quantidade de recurso fora das faixas impostas, seja pela solicitação por recursos do experimento, ou pelas instruções *guarantee_resource*.

Os conflitos de consumo de recursos são verificados de duas formas diferentes, de acordo com o tipo de recurso. Recursos do tipo ‘reservável’ são verificados apenas somando todos os consumos, por todos os objetos, no momento-chave.

Recursos do tipo ‘consumível’ são tratados de forma diferente, devido à abordagem da RASSO_ml para a modelagem de seu consumo. O cálculo da quantidade de recursos consumidos é feito multiplicando-se cada taxa de consumo pelo tempo decorrido desde que a taxa foi definida (por exemplo, no momento da execução de uma ação *Turn_On*).

A partir daí, os consumos são somados e verificados da mesma forma que ocorre com os recursos reserváveis.

Um conflito no consumo de um recurso consumível detectado em um momento-chave não necessariamente surgiu naquele momento. Como o consumo é informado em termos de taxas, o estouro na quantidade consumida pode ter ocorrido em qualquer instante entre o momento-chave atual e o momento-chave imediatamente anterior a ele. É necessário então determinar o momento exato do estouro do consumo.

Para isso é aplicado um algoritmo, baseado em busca binária, ao período entre os dois momentos-chave (o anterior, quando ainda não havia conflito, e o atual, quando o conflito foi detectado). O instante exato do estouro do consumo é importante para que o planejador possa deslocar temporalmente, de forma mais eficaz, as ações no esboço do plano.

Sabendo quais conflitos ocorrem, seus tipos e em que instantes, o algoritmo identifica quais ações contribuem para o conflito. Para conflitos de estados, os contribuintes são todas as ações que modificam aquele estado antes do momento-chave que está sendo verificado. Para conflitos de recursos, os contribuintes são todas as ações que modificam o perfil de consumo do recurso antes do momento-chave.

6.4.3 O Teste e a Aplicação de Mudanças no Esboço de Plano

O próximo passo do algoritmo de planejamento é tentar modificar o plano, de forma a resolver os conflitos identificados em determinado momento-chave. Para cada conflito, é testado um conjunto de alterações no plano que possam resolvê-lo. São testadas modificações para cada uma das ações que contribuem para o conflito. Estas modificações podem ser:

- Mover temporalmente no plano a ação que contribui para o conflito. Esta é a principal modificação testada, pois é a que provavelmente irá gerar menor impacto no plano;

- Adicionar outra ação no plano, em um momento aleatório antes da ação que contribui para o conflito;

- Excluir do plano a ação que contribui para o conflito.

Cada uma das modificações testadas é avaliada. A modificação considerada a ‘melhor’ é então aplicada ao esboço de plano. A melhor modificação é aquela que resolve o conflito que está sendo tratado, e insere o menor número possível de conflitos no plano (o ideal é não inserir novos conflitos). Outros critérios são utilizados para ‘desempate’ na seleção de modificações, como o tipo de conflitos adicionados e suas durações.

Caso não seja encontrada nenhuma mudança no plano que resolva determinado conflito, uma perturbação é inserida no plano. A perturbação consiste da adição de uma ação que resolva o conflito (se houver tal ação), mas cujos pré-requisitos não sejam verdadeiros no momento em que ela foi inserida. A falta dos pré-requisitos da ação adicionada insere novos conflitos no plano. Para determinar onde é permitido ao planejador inserir uma perturbação, é utilizada na descrição do modelo a instrução *force_state_if_needed_by_id*, (veja a Figura 6.4).

Após a seleção e aplicação de uma modificação no esboço de plano, um novo levantamento dos conflitos é feito. Este processo segue até que não haja mais conflitos no plano. Quando isso ocorrer, o plano obtido é enviado para a fila de telecomandos temporizados ou disponibilizado para telemetria, de acordo com o modo de planejamento do RASSO, conforme descrito no item 5.4 deste trabalho.

Cabe frisar aqui que, ao contrário do que era esperado no início deste trabalho, não foi o algoritmo de planejamento que demandou maior volume de trabalho, e sim a definição de uma forma adequada para a representação do conhecimento do domínio de satélites, que culminou na criação da RASSO_ml. A representação correta do modelo, bem como do problema a ser resolvido, tornou mais simples o processo de busca pela solução no algoritmo de planejamento.

O próximo Capítulo apresenta o cenário de testes criado para validar o sistema, e os resultados obtidos de sua execução.

CAPÍTULO 7

CENÁRIO DE TESTES E RESULTADOS OBTIDOS

Este Capítulo descreve o comportamento assumido para os experimentos embarcados no EQUARS, o cenário criado para os testes do protótipo do RASSO, e apresenta os resultados da execução do replanejamento, obtidos a partir da simulação da solicitação por mais recursos, disparada por um dos experimentos.

São então descritos o processo de replanejamento e as escolhas feitas pelo planejador, com os conflitos resolvidos e inseridos durante o processo. A seguir, apresenta-se o tempo gasto com a composição do problema e com o planejamento, bem como o tamanho da aplicação compilada, mostrando que ela está adequada para execução a bordo de satélites, com o computador COMAV.

7.1 O Modelo do Satélite EQUARS

Este item apresenta o modelo criado para o satélite EQUARS, composto de suas descrições estática e dinâmica. O modelo em RASSO_ml correspondente à descrição aqui colocada encontra-se no Apêndice A deste trabalho.

A descrição estática do satélite EQUARS em RASSO_ml compreende:

- Duas classes, *Experiment* e *Antenna*;
- Cinco objetos da classe *Experiment* (CERTO, GROM, IONEX, MLTM e TIP), e um da classe *Antenna* (*Comm_System*);
- Dois recursos: *Power*, do tipo *reservable*, e *Mass_Memory*, do tipo *depletable*.

Objetos da classe *Experiment* possuem atributos para nome, *status* (ligado/desligado), modo de operação, taxa de aquisição de dados, precisão dos dados coletados e

prioridade de execução com relação aos demais. A classe *Antenna* possui apenas um atributo, que indica se o sistema de comunicação está ligado ou desligado.

O recurso *Power* representa a bateria do satélite e é limitado ao fornecimento de 50 Watts de energia, enquanto que *Mass_Memory* representa a memória de massa para os dados dos experimentos, e disponibiliza 120 Mbits a serem compartilhados entre todos eles.

A descrição dinâmica do modelo é composta por:

- Cinco ações: *Turn_On*, *Turn_Off*, *Change_Mode*, *Change_Sample_Rate* e *Change_Precision*;
- Três janelas de tempo: *Day*, *Night* e *Communicating*;
- Três comportamentos, um associado a cada janela de tempo: *When_Day*, *When_Night* e *When_Communicating*.

As ações não são aplicáveis a todos os objetos. *Turn_On* e *Turn_Off*, por exemplo, não podem ser executadas para o experimento IONEX. Este experimento não pode ser desligado, pode-se apenas alternar seus modos de operação entre *full* e *partial* pela ação *Change_Mode*. O IONEX está plenamente ativo durante a fase em eclipse da órbita, e parcialmente ativo durante a fase iluminado. *Change_Sample_Rate* e *Change_Precision* alteram, respectivamente, a taxa de amostragem e a precisão dos dados coletados por um experimento, e aumentam o consumo de memória e energia para o experimento.

As janelas de tempo representam as fases iluminada e em eclipse de uma órbita, e o período de comunicação com a estação de rastreio em solo. As duas primeiras janelas são consecutivas, e a terceira pode ocorrer a qualquer momento, sobrepondo-se às anteriores.

O comportamento associado à janela *Day* indica que o experimento MLTM é ligado automaticamente ao entrar na fase iluminada da órbita, e desligado ao sair dela. Da

mesma forma, o comportamento para *Night* indica que CERTO é ligado ao entrar na fase em eclipse, e desligado ao término desta fase.

O comportamento para a janela de comunicação com solo descreve a operação do sistema de comunicação do satélite. Ao ser ligado, este sistema passa a consumir energia e 'gerar' memória. Esta geração deve-se ao fato de que os dados coletados pelos experimentos estão sendo enviados para solo, liberando assim a memória utilizada para seu armazenamento. Ao término da janela, a 'geração' de memória e o consumo de energia pelo sistema de comunicação são encerrados.

Como pode ser constatado no Apêndice A, foram adicionadas restrições ao planejador nos comportamentos através de instruções *guarantee_resource*. Isso impede que o planejador modifique o plano de forma a retirar recursos de determinados experimentos, durante a ocorrência das janelas de tempo.

7.2 O Cenário de Testes

O cenário criado para os testes do RASSO engloba um período de três órbitas. Foi assumido que cada órbita tem a duração de 6000 segundos (100 minutos), com fases 'Dia' e 'Noite' de 3000s (50 minutos) cada. A duração de cada janela de comunicação, que ocorre uma vez a cada órbita, é de 540s (9 minutos). O momento de início do cenário é o momento zero.

O experimento TIP, conforme apresentado no item 5.1.2 deste trabalho, é um imageador de luminescência para relâmpagos e *sprites*, que são fenômenos que possuem como característica a curta duração e a aleatoriedade da ocorrência. Por este motivo, o TIP foi escolhido para enviar a solicitação por mais recursos ao RASSO.

Foi criado um processo do aplicativo do RTEMS para simular o disparo da solicitação por recursos. A simulação da detecção da ocorrência do fenômeno se dá logo aos 3 segundos (a partir do momento zero), e são solicitados mais energia e memória (quantidades três vezes maiores do que em operação normal) para o TIP por um período de 3000s, a partir dos 9000s do início do plano.

Foi definido um plano de operações inicial e uma tabela de ocorrências de janelas de tempo para o cenário, que são apresentados na Figura 7.1.

Fila de Telecomandos Temporizados

Momento	Comando
0	Change_Mode(ionex, partial)
600	Turn_On(grom)
1800	Turn_Off(grom)
3000	Change_Mode(ionex, (full)
3000	Turn_On(tip)
3600	Turn_On(grom)
4800	Turn_Off(grom)
6000	Change_Mode(ionex, partial)
6000	Turn_Off(tip)
6600	Turn_On(grom)
7800	Turn_Off(grom)
9000	Change_Mode(ionex, full)
9000	Turn_On(tip)
9600	Turn_On(grom)
10800	Turn_Off(grom)
12000	Change_Mode(ionex, partial)
12000	Turn_Off(tip)
12600	Turn_On(grom)
13800	Turn_Off(grom)
15000	Change_Mode(ionex, (full)
15000	Turn_On(tip)
15600	Turn_On(grom)
16800	Turn_Off(grom)

Tabela de Janelas de Tempo

Janela	inicia em	termina em
Day	0	3000
Night	3000	6000
Communicating	4200	4740
Day	6000	9000
Night	9000	12000
Communicating	10200	10740
Day	12000	15000
Night	15000	18000
Communicating	16200	16740

*** todas as referências temporais estão em segundos.*

FIGURA 7.1 - Comandos Iniciais e Janelas de Tempo Definidos para o Cenário

Na execução do plano normal, o consumo de recursos mantém-se dentro dos limites impostos. Entretanto, caso fosse atendida sem nenhuma outra alteração na operação do satélite, a solicitação por mais recursos enviada pelo TIP faria com que o consumo, tanto de memória quanto de energia, ultrapassasse as quantidades disponíveis no satélite. O consumo de energia seria colocado em sobrecarga aos 10800s, e o de memória, aos 15600s.

A Figura 7.2 apresenta o consumo normal e em sobrecarga de cada recurso. Nela, as linhas vermelhas indicam a quantidade máxima disponível para consumo.

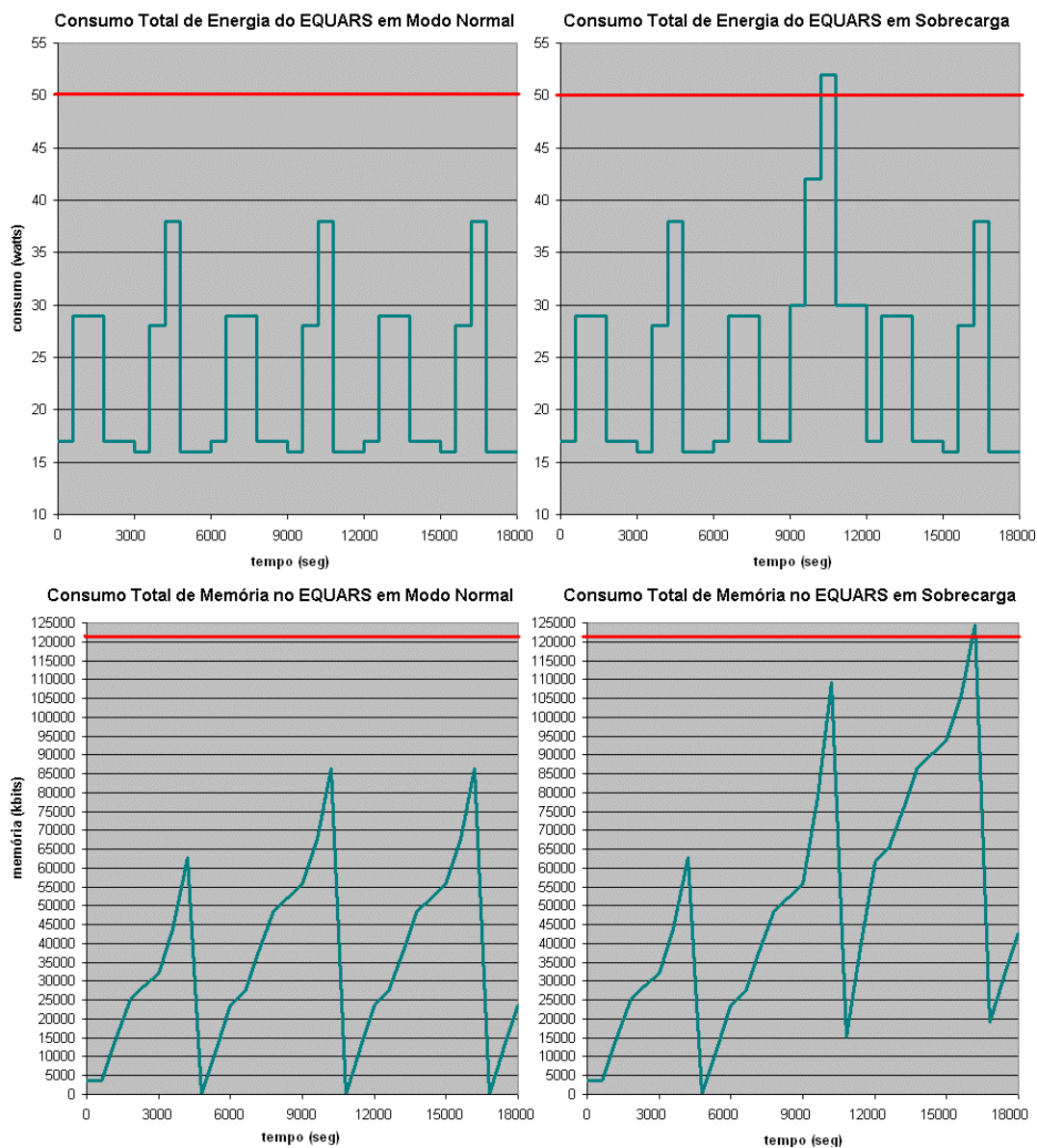



FIGURA 7.2 - Consumo de Energia e Memória em Modo Normal e em Sobrecarga

Cabe ao RASSO trabalhar no plano e encontrar alterações nele que permitam atender à solicitação do TIP, sem colocar os recursos em sobrecarga, e respeitando as restrições impostas no modelo do EQUARS. O próximo tópico apresenta, passo a passo, as escolhas feitas pelo Planejador durante o processo de replanejamento.

7.3 O Processo de Replanejamento e as Escolhas do Planejador

Ao receber o esboço de plano do Compositor de Problemas, o Planejador tinha como únicos conflitos presentes neste plano os de consumo de energia e memória pelo TIP, no período indicado pela solicitação de recursos: entre 9000s e 12000s.

Ao iniciar a busca, o Planejador detectou que a melhor forma de resolver este conflito seria modificar a taxa de amostragem do experimento, inserindo no plano uma ação *Change_Sample_Rate(tip, 3)* aos 9000s. Esta ação multiplica em três vezes o consumo de energia e memória pelo TIP, atendendo à solicitação do experimento.

Entretanto, conforme esperado, a inserção desta ação levou a conflitos de consumo total de energia, aos 10800s, e de memória, aos 15600s (veja a Figura 7.2). Foi também inserido um conflito de estado imposto, pois TIP deveria ter sua taxa de amostragem igual a '1' ao término do período do plano (horizonte h3), e *Change_Sample_Rate* alterou esta taxa, tornando-a diferente do esperado em h3. O Planejador passou então a buscar mais uma modificação no plano, que solucionasse estes novos conflitos.

A modificação encontrada foi inserir um comando para desligar o experimento GROM, aos 10143s (um momento aleatório que foi testado pelo Planejador). Esta modificação resolveu os conflitos de consumo total de energia e memória, deixando apenas o conflito de estado imposto ao TIP para ser resolvido.

Em uma nova busca por modificações que resolvam este conflito, o Planejador não encontrou opções aplicáveis. Ele inseriu então uma perturbação no plano para tentar solucionar o conflito. Como já explicado, a perturbação consiste da adição de uma ação que resolva o conflito, mas cujos pré-requisitos não sejam verdadeiros no momento em que ela foi inserida.

A ação adicionada como perturbação no plano foi a *Change_Sample_Rate(tip, 1)*, inserida aos 12000s. Apesar de esta ação devolver a taxa de amostragem do TIP ao seu normal, após o período de observação do fenômeno, ela tem como pré-requisito que o TIP esteja ligado, o que era falso aos 12000s (havia uma ação *Turn_Off* imediatamente

antes). Um novo conflito de estado imposto foi inserido ('TIP deveria estar ligado aos 12000s'), e o Planejador buscou uma ação para ligar o TIP.

Após inserir *Turn_On(tip)* e resolver mais este conflito, um último conflito foi inserido: TIP deveria estar novamente desligado após os 12000s. A adição de um novo comando *Turn_Off(tip)*, também aos 12000s solucionou o problema, sem adicionar novos conflitos.

O plano modificado para atender à solicitação de TIP é apresentado na Figura 7.3. Os comandos inseridos pelo RASSO estão em negrito.

Fila de Telecomandos Temporizados

Momento	Comando
0	Change_Mode(ionex, partial)
600	Turn_On(grom)
1800	Turn_Off(grom)
3000	Change_Mode(ionex, (full)
3000	Turn_On(tip)
3600	Turn_On(grom)
4800	Turn_Off(grom)
6000	Change_Mode(ionex, partial)
6000	Turn_Off(tip)
6600	Turn_On(grom)
7800	Turn_Off(grom)
9000	Change_Mode(ionex, full)
9000	Turn_On(tip)
9000	**Change_Sample_Rate(tip, 3)**
9600	Turn_On(grom)
10143	**Turn_Off(grom)**
10800	Turn_Off(grom)
12000	Change_Mode(ionex, partial)
12000	Turn_Off(tip)
12000	**Turn_On(tip)**
12000	**Change_Sample_Rate(tip, 1)**
12000	**Turn_Off(tip)**
12600	Turn_On(grom)
13800	Turn_Off(grom)
15000	Change_Mode(ionex, (full)
15000	Turn_On(tip)
15600	Turn_On(grom)
16800	Turn_Off(grom)

FIGURA 7.3 - O Plano de Operações Modificado Pelo RASSO

A Figura 7.4 apresenta o consumo total de memória e energia, conforme definido no plano modificado pelo RASSO.

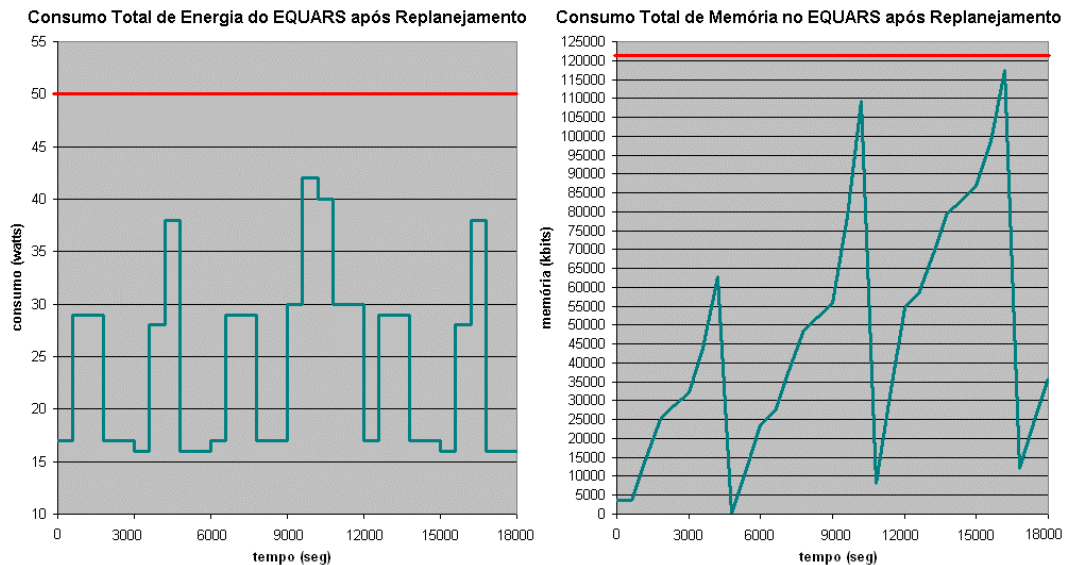



FIGURA 7.4 - Consumo Total de Energia e Memória Após o Replanejamento

Nota-se que, além de evitar a sobrecarga no consumo de recursos, o RASSO garantiu que, ao término do período do plano (aos 18000s), as quantidades em uso de recursos foram iguais (energia) ou menores (memória) do que no plano original. Isso, mais a imposição dos estados em que os experimentos devem estar ao término do período, garantem que os efeitos da modificação do plano sejam temporários, e que, após este período, o satélite volte a operar conforme originalmente planejado.

7.4 Análise do Processo de Replanejamento

Embora tenha atingido todos os objetivos colocados no cenário, pode-se ver pela Figura 7.3 que o plano de operações modificado poderia ser melhor.

O comando para o desligamento de GROM aos 10143 segundos apenas 'adianta' o desligamento do experimento, originalmente previsto para os 10800 segundos. Foi notado que, apesar de dispor de opções para mover e excluir ações, o Planejador sempre

decidiu que a inserção de ações teria melhor resultado. No caso do desligamento de GROM, teria sido mais ‘inteligente’ mover a ação já existente no plano.

Outra melhoria possível diz respeito aos comandos inseridos aos 12000s para o ajuste de TIP ao estado que lhe foi imposto ao término do plano. O planejador não conseguiu detectar que, caso *Change_Sample_Rate(tip, 1)* fosse inserido antes do comando existente no plano original para desligar TIP aos 12000s, não teria sido necessário inserir novos comandos para religá-lo e desligá-lo novamente neste mesmo segundo.

Estas limitações na solução encontrada podem ser resolvidas futuramente com a inserção de heurísticas no Planejador.

7.5 Adequação do RASSO ao COMAV

O cenário apresentado no item anterior foi executado em uma placa de desenvolvimento com o processador ERC32, de propriedade do Grupo de Supervisão de Bordo (SUBORD), do INPE. O processador foi configurado para trabalhar a uma velocidade de 12MHz, pois esta provavelmente será a velocidade utilizada no COMAV.

O processo de replanejamento descrito nos itens anteriores levou 114,28 segundos para ser executado, fazendo uso total do tempo do processador. Deste período, a inicialização do sistema, do modelo e a composição do problema levaram apenas 0,07 segundo para serem completados.

O executável do protótipo do RASSO, com seu modelo e o sistema operacional RTEMS inclusos, ocupa 182,68 kbytes em memória de programa. Descontando o RTEMS, que faz parte do COMAV, o RASSO e seu modelo compilado possuem 79,96 kbytes de tamanho.

O tempo de replanejamento e o tamanho da aplicação mostram que o protótipo do RASSO está adequado para execução em computadores de bordo dos satélites do INPE.

A implementação do protótipo do RASSO gerou 3626 linhas de código-fonte (com comentários, descontadas as linhas em branco), das quais 913 compõem a definição da linguagem RASSO_ml e 321 fazem parte do modelo do satélite.

CAPÍTULO 8

CONCLUSÃO

O desenvolvimento de um protótipo de serviço de replanejamento embarcado pode ser considerado um primeiro passo no sentido de aumentar a autonomia do *software* embarcado nos futuros satélites do INPE.

O protótipo desenvolvido neste trabalho traz os conceitos de planejamento e escalonamento da área de IA para os sistemas embarcados em satélites do INPE, de forma a permitir o replanejamento autônomo de operações em resposta a eventos externos. Isso dá ao satélite maior poder de resposta, e possibilita o uso otimizado de seus recursos.

A seguir são apresentadas as principais contribuições deste trabalho, os trabalhos futuros vislumbrados e outras aplicações possíveis para esta tecnologia, dentro do INPE.

8.1 Principais Contribuições

Diversas contribuições podem ser destacadas neste trabalho, sendo que as principais seguem listadas abaixo.

Para o INPE, este trabalho contribui trazendo um primeiro estudo da aplicação de técnicas de Planejamento e Escalonamento em IA embarcadas em satélites, dentro do contexto específico do Instituto. Isso possibilita a análise da viabilidade do uso destas técnicas, seja para a aplicação em satélites científicos, seja para qualquer outra aplicação embarcada em satélites.

Nas pesquisas realizadas para esta Dissertação, não foi encontrado na literatura nenhum trabalho que reunisse informações sobre os sistemas planejadores com IA na área espacial. Todos os dados encontrados vieram de artigos descrevendo sistemas específicos, contendo no máximo comparações pontuais com um ou dois outros

planejadores. Para aqueles interessados no estudo da automatização e autonomia de missões espaciais com o uso de Planejamento e Escalonamento em IA, o Capítulo 4 apresenta uma revisão dos sistemas desenvolvidos e em desenvolvimento na área, assim como das atuais tendências.

Para os envolvidos com planejamento em IA na área espacial, a linguagem desenvolvida para o protótipo traz uma nova forma de representar o consumo de recursos, especialmente adequada para uso no segmento espacial, por estar mais próxima à forma de operação real de satélites e espaçonaves em geral.

Para a comunidade de planejamento e escalonamento em IA, este trabalho representa uma contribuição significativa para a representação de problemas de planejamento. O fato de não ser possível utilizar PDDL ou qualquer outra linguagem de representação do conhecimento no ambiente de execução do RASSO levou à criação da RASSO_ml, que possui uma série de características bastante diferentes do que é comumente visto em sistemas baseados nos conceitos da STRIPS e da PDDL:

- A integração do modelo ao código-fonte;
- O uso dos recursos da linguagem de programação para aumentar a representatividade do modelo;
- A forma de descrever ações e eventos exógenos e as instruções para manipulação de objetos no tempo;
- A aplicação de *structs* para a descrição de objetos, o que leva o modelo mais próximo à Orientação a Objetos do que, por exemplo, a OCL (que, conforme mostrado no Capítulo 3, toma por 'objeto' um agrupamento de predicados). O uso de OO para descrever modelos de domínio para planejamento é algo que vem sendo perseguido pela comunidade nos últimos anos.

A aceitação de artigos relacionados a este trabalho em conferências internacionais mostra o interesse da comunidade nas idéias aqui implementadas. Um artigo apresentando uma visão inicial deste trabalho foi publicado na *9th International*

Conference on Space Operations (SpaceOps 2006) (Kucinskis and Ferreira, 2006), e selecionado posteriormente entre os melhores trabalhos do congresso para publicação como capítulo em uma edição futura de um livro da série *Progress in Astronautics and Aeronautics*, prevista para publicação no segundo semestre deste ano.

Outro artigo foi publicado e apresentado na *2007 IEEE Aerospace Conference* (Kucinskis et al., 2007), no início de março deste ano. Outros trabalhos foram submetidos para os *journals Applied Artificial Intelligence*, *Journal of Aerospace Computing, Information, and Communication (JACIC)* e *Engineering Applications of AI*, e se encontram atualmente em processo de revisão.

8.2 Trabalhos Futuros

Há muito que ser feito com relação a estudos e implementação de técnicas de planejamento e escalonamento em IA para o segmento espacial.

Em primeiro lugar, destaca-se a necessidade do desenvolvimento de técnicas de verificação e validação para *software* autônomo. Discussões a este respeito e algumas propostas interessantes podem ser vistas em (Brat et al., 2006) e (Blanquart et al., 2004).

O gerenciamento da incerteza em sistemas autônomos é um tópico ainda pouco explorado, e de vital importância para o futuro da autonomia em sistemas espaciais. O CLARATy, um dos sistemas citados no Capítulo 4 deste trabalho, se propõe a realizar isso. Maiores informações são encontradas em (Bresina et al., 2002).

O protótipo do RASSO também traz muitas possibilidades de melhorias em trabalhos futuros. A aplicação de heurísticas específicas para o domínio pode melhorar consideravelmente o desempenho do sistema. A linguagem RASSO_ml também possui diversas melhorias possíveis. Outra possibilidade é o desenvolvimento de ferramentas de *software* a serem utilizadas em solo pelos operadores da missão, para auxiliar na análise dos planos modificados em bordo e transmitidos pelo satélite.

Finalmente, uma migração do RASSO para C++ possibilitaria o uso de objetos, ao invés de *structs*, na modelagem do domínio. Isso permitiria o uso real de Orientação a Objetos na linguagem de representação do conhecimento.

Apesar da aplicação escolhida para este trabalho ter sido a resposta à detecção de fenômenos de curta duração em satélites científicos, há um grande número de outras aplicações para as técnicas aqui implementadas. Entre elas, destacam-se duas de especial interesse para o INPE:

- O uso de replanejamento em satélites de sensoriamento remoto. Um satélite capaz de efetuar replanejamento embarcado pode, em tempo real, modificar seu plano para obter mais e melhores imagens de uma queimada detectada na Amazônia, por exemplo;

- A descoberta de novas formas de operação em modos degradados de funcionamento. Com o passar do tempo, os equipamentos dos satélites apresentam falhas, e a forma de operá-los deve ser adaptada para lidar com estas falhas. Um sistema de replanejamento, embarcado em um satélite que apresente falhas, pode ajudar a equipe em solo a descobrir novas formas de operar o satélite. Para isso, ele receberia como objetivos fazer o mesmo que faria se não apresentasse falha alguma. As falhas seriam então inseridas no problema como restrições impostas, o que forçaria o planejador a buscar formas alternativas para atingir seus objetivos.

8.3 Considerações Finais

É difícil mudar a visão predominante na área técnica, na qual se espera total previsibilidade dos sistemas embarcados, para uma visão que aceite o comportamento de um sistema autônomo. A realização de estudos e o desenvolvimento de protótipos com o objetivo de aumentar a autonomia de satélites permitem uma melhor compreensão das técnicas utilizadas, com seus pontos fortes e fracos.

A melhor compreensão destas técnicas leva a uma definição clara dos limites a serem impostos para sistemas autônomos, e em que nível a autonomia pode ser aplicada ao ambiente espacial.

REFERÊNCIAS BIBLIOGRÁFICAS

AARUP, M.; ARENTOFT, M. M.; PARROD, Y.; STADER, J.; STOKES, I. Optimum-AIV: a knowledge-based planning and scheduling system for spacecraft AIV. **Knowledge Based Scheduling,** Fox, M. and Zweben, M., ed.. Morgan Kaufmann, San Mateo, CA, USA, 1994.

AGÊNCIA ESPACIAL BRASILEIRA (AEB). **PNAE: programa nacional de atividades espaciais 2005-2014.** Brasília: MCT/AEB, 2005. Disponível em: <http://www.aeb.gov.br/area/download/pnae_web.pdf>. Acesso em 14 mar. de 2007.

AI-CHANG, M.; BRESINA, J.; CHAREST, L.; CHASE, A.; CHENG-JUNG, J.; JONSSON, A.; KANEFSKY, B.; MORRIS, P.; RAJAN, K.; YGLESIAS, J.; CHAFIN, B.; DIAS, W.; MALDAGUE, P. MAPGEN: mixed-initiative planning and scheduling for the Mars Exploration Rover mission. **IEEE Intelligent Systems**, v. 19, n. 1, pp. 8-12, Jan/Feb 2004.

ALONSO, J. D. D.; PEREIRA JR., A. C. O.; PESSOTTA, F. A. **Especificação de requisitos do COMAV.** São José dos Campos: DEA/INPE, 2001.

AXMANN, R.; WICKLER, M. Development and verification of an autonomous on-board mission planning system – an example from the BIRD satellite. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 9., 2006, Rome, Italy. **Proceedings...** Rome: AIAA, 2006.

BEEK, P.; CHEN, X. CPlan: a constraint programming approach to planning. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-99), 16., 1999, Orlando, FL, USA. **Proceedings...** Orlando: AAAI Press, 1999. p. 585-590.

BERNARD, D.; DORAIS, G.; GAMBLE, E.; KANEFSKY, B.; KURIEN, J.; MAN, G. K.; MILLAR, W.; MUSCETTOLA, N.; NAYAK, P.; RAJAN, K.; ROUQUETTE, N.; SMITH, B.; TAYLOR, W.; TUNG, Y. Spacecraft autonomy flight experience: the DS1 Remote Agent Experiment. In: SPACE TECHNOLOGY CONFERENCE AND EXPOSITION, 1999, Albuquerque, NM, USA. **Proceedings...** Albuquerque: AIAA, 1999.

BIANCHO, A. C.; AQUINO, A. C.; FERREIRA, M. G. V.; SILVA, J. D. S.; CARDOSO, L. A multi-agent ground-operations automation architecture. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 9., 2006, Rome, Italy. **Proceedings...** Rome: AIAA, 2006.

BIUNDO, S.; AYLETT, R.; BEETZ, M.; BORRAJO, D.; CESTA, A.; GRANT, T.; MCCLUSKEY, L.; MILANI, A.; AND VERFAILLE, G. (Ed.) **Technological roadmap on AI planning and scheduling.** PLANET II: The European Network of Excellence in AI Planning, IST-2000-29656, 2003. Disponível em: <http://planet.dfki.de/service/Resources/Roadmap/Roadmap2.pdf>.

BLANQUART, P.; FLEURY, S.; HERNEK, M.; HONVAULT, C.; INGRAND, F.; PONCET, J. C.; POWELL, D.; STRADY-LÉCUBIN, N.; THÉVENOD-FOSSE, P. Software safety supervision on-board autonomous spacecraft. In: EUROPEAN CONGRESS ON EMBEDDED REAL TIME SOFTWARE (ERTS), 2., 2004, Toulouse, France. **Proceedings...** [s.n.], 2004. 11p.

BLUM, A.; FURST, M. Fast planning through planning graph analysis**.** In: INTERNATIONAL JOINT CONFERENCES ON ARTIFICIAL INTELLIGENCE (IJCAI-95), 1995, Montreal, Canada. **Proceedings...** Montreal: Morgan Kaufmann, 1995, p. 1636–1642.

BONET, B.; LOERINCS, G.; GEFFNER, H. A robust and fast action selection mechanism for planning. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-97), 14., 1997, Providence, RI, USA. **Proceedings...** Providence: AAAI/MIT Press, 1997. pp. 714-719.

BONET, B.; GEFFNER, H. HSP: planning as heuristic search. In: **Entry at the AIPS-98 Planning Competition**, Pittsburgh, USA, Jun 1998.

BRAT, G.; DENNEY, E.; FARELL, K.; GIANNAKOPOULOU, D.; JONSSON, A.; FRANK, J.; BODDY, M.; CARPENTER, T.; ESTLIN, T.; PIVTORAIKO, M. A robust compositional architecture for autonomous systems. In: IEEE AEROSPACE CONFERENCE, 2006, Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2006.

BRESINA, J.; DEARDEN, R.; MEULEAU, N.; RAMAKRISHNAN, S.; SMITH, D.; WASHINGTON, R. Planning under continuous time and resource uncertainty: a challenge for AI. In: Conference on Uncertainty in AI (UAI), 18., 2002, Edinburgh, Scotland. **Proceedings...** [s.n.], 2002.

BRESINA, J.; JONSSON, A.; MORRIS, P.; RAJAN, K. Activity planning for the Mars Exploration Rovers. In: INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING (ICAPS05), 2005, Monterey, CA, USA. **Proceedings...** Monterey: AIAA, 2005. pp. 40–49.

BRESINA, J. L.; MORRIS, P.H. Mission operations planning: beyond MAPGEN. In: IEEE INTERNATIONAL CONFERENCE ON SPACE MISSION CHALLENGES FOR INFORMATION TECHNOLOGY, 2., 2006, Pasadena, CA, USA. **Proceedings...** Pasadena: IEEE, 2006.

CARDOSO, L. S.; FERREIRA, M. G. V.; ORLANDO, V. An intelligent system for generation of automatic flight operation plans for the satellite control activities at INPE. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 9., 2006, Rome, Italy. **Proceedings...** Rome: AIAA, 2006.

CARDOSO, L. S. **Aplicação da tecnologia de agentes de planejamento em operações de satélites.** 2006. Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2006.

CARVALHO, H. **EQUARS system architecture.** São José dos Campos: INPE. EQUARS Workshop, Nov 2001.

CESTA, A.; CORTELLESSA, G.; ODDI, A.; POLICELLA. N. Studying decision support for MARS EXPRESS planning tasks: a report from the MEXAR experience. In: INTERNATIONAL WORKSHOP ON PLANNING AND SCHEDULING FOR SPACE, 4., 2004, Darmstadt, Germany. **Proceedings...** Darmstadt: ESA-ESOC, 2004.

CESTA, A.; ODDI, A.; CORTELLESSA, G.; FRATINI, S.; POLICELLA, N. AI based tools for continuous support to mission planning. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 9., 2006, Rome, Italy. **Proceedings...** Rome: AIAA, 2006.

CHIEN, S.; MUSCETTOLA, N.; RAJAN, K.; SMITH, B.; RABIDEAU, G. Automated planning and scheduling for goal-based autonomous spacecraft. **IEEE Intelligent Systems**, v. 13, n. 5, pp. 50-55, 1998.

CHIEN, S.; KNIGHT, R.; STECHERT, A.; SHERWOOD, R.; RABIDEAU, G. Using iterative repair to increase the responsiveness of planning and scheduling for autonomous spacecraft. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI), 16., 1999, Stockholm, Sweden. **Proceedings...** Stockholm: Morgan Kaufmann, 1999.

CHIEN, S.; ENGELHARDT, B.; KNIGHT, R.; RABIDEAU, G.; SHERWOOD, R.; HANSEN, E.; ORTIVIZ, A.; WILKLOW, C.; WICHMAN, S. Onboard autonomy on the Three Corner Sat mission. In: INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE ROBOTICS AND AUTOMATION IN SPACE (I-SAIRAS), 6., 2001, Montreal, Canada. **Proceedings...** Montreal: CSA, 2001.

CHIEN, S.; SHERWOOD, R.; TRAN, D.; CASTANO, R.; CICHY, B.; DAVIES, A.; RABIDEAU, G.; TANG, N.; BURL, M.; MANDL, D.; FRYE, S.; HENGEMIHLE, J.; D'AGOSTINO, J.; BOTE, R.; TROUT, B.; SHULMAN, S.; UNGAR, S.; VAN GAASBECK, J.; BOYER, D.; GRIFFIN, M.; BURKE, H.; GREELEY, R.; DOGGETT, T.; WILLIAMS, K.; BAKER, V.; DOHM, J. Autonomous science on the EO-1 mission. In: INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE ROBOTICS AND AUTOMATION IN SPACE (I-SAIRAS), 7., 2003, Nara, Japan. **Proceedings...** Nara: JAXA, 2003.

CHIEN, S.; SHERWOOD, R.; TRAN, D.; CICHY, B.; RABIDEAU, G.; CASTANO, R.; DAVIES, A.; LEE, R.; MANDL, D.; FRYE, S.; TROUT, B.; D'AGOSTINO, J.; SHULMAN, S.; BOYER, D.; HAYDEN, S.; SWEET, A.; CHRISTA, S. Lessons learned from Autonomous Sciencecraft Experiment. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS '05), 4., 2005, Utrecht, Netherlands. **Proceedings...** Utrecht: ACM Press, 2005.

CONSULTATIVE COMMITTEE FOR SPACE DATA SYSTEMS (CCSDS). **Telemetry – summary of concept and rationale.** Washington, D.C., USA, December 1987. Report Concerning Space Data Systems Standards, CCSDS 100.0-G-1. Green Book. Issue 1.

CONSULTATIVE COMMITTEE FOR SPACE DATA SYSTEMS (CCSDS). **Telecommand – summary of concept and rationale.** Washington, D.C., USA, December 1987. Report Concerning Space Data Systems Standards, CCSDS 200.0-G-6. Green Book. Issue 6.

COOK, S. A.; MITCHELL, D. G. Finding hard instances of the satisfiability problem: a survey. In: **Satisfiability Problem: Theory and Applications**, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1997.

CURRIE, K.; TATE, A. O-Plan: the open planning architecture. **Artificial Intelligence**, v. 52, n. 1, pp 49-86, Elsevier, Nov 1991.

DAMIANI, S.; VERFAILLIE, G.; CHARMEAU, M. C. An Earth watching satellite constellation: how to manage a team of watching agents with limited communications. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS '05), 4., 2005, Utrecht, The Netherlands. **Proceedings...** Utrecht: ACM Press, 2005.

DRABBLE, B. Mission scheduling for spacecraft: the diaries of T-SCHED. In INTERNATIONAL EXPERT PLANNING SYSTEMS CONFERENCE, 1., 1990, London, England. **Proceedings...** Brighton: Institute of Electrical Engineers, 1990.

EROL, K.; NAU, D. S.; SUBRAHMANIAN, V. S. Complexity, decidability and undecidability results for domain-independent planning. Artificial Intelligence, v. 76, n. 1-2, pp. 75-88, Elsevier Science, Jul 1995.

ESTLIN, T.; GAINES, D.; CHOUINARD, C.; FISHER, F.; CASTANO, R.; JUDD, M.; ANDERSON, R.; NESNAS, I. Enabling autonomous rover science through dynamic planning and scheduling. In: IEEE AEROSPACE CONFERENCE, 2005, Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2005.

FIKES, R. E.; NILSSON, N. J. STRIPS: a new approach to the application of theorem proving to problem-solving. **Artificial Intelligence**, v. 2, n. 3-4, pp. 189-208, 1971.

FRANK, J.; JONSSON, A. K. Constraint-based attribute and interval planning. **Constraints**, v. 8, n. 4, pp 339-364, Oct 2003.

FRATINI, S.; CESTA, A. A planning and scheduling tool to automate and support mission planning. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 9., 2006, Rome, Italy. **Proceedings...** Rome: AIAA, 2006.

FUCHS, J.; GULDBERG, J.; OLALAINTY, B.; DARROY, J. M.; CURRIE, K. **Final report, expert planning systems study.** Technical Report, EP-CRI-FR-0001-1988, CRI, 1988.

FUCHS, J. J.; GASQUET, A.; OLALAINTY, B.; CURRIE, K. W. PlanERS-1: an expert planning system for generating spacecraft mission plans. In INTERNATIONAL EXPERT PLANNING SYSTEMS CONFERENCE, 1., 1990, London, England. **Proceedings...** Brighton: Institute of Electrical Engineers, 1990. pp. 70-75.

FUKUNAGA, A.; RABIDEAU, G.; CHIEN, S.; YAN, D. Towards an application framework for automated planning and scheduling. In: INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE ROBOTICS AND AUTOMATION IN SPACE (I-SAIRAS), 4., 1997, Tokyo, Japan. **Proceedings...** Tokyo: NASDA, 1997.

HALSEY, K.; LONG, D.; FOX, M. CRIKEY - a temporal planner looking at the integration of scheduling and planning. In: INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING (ICAPS04), 2004, Whisler, Canada. **Proceedings...** Whisler: AIAA, 2004.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Homepage da MECB - missão espacial completa brasileira.** São José dos Campos, 1999. Disponível em <http://www.inpe.br/programas/mecb/default.htm>. Acesso em 17 de janeiro de 2007.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Homepage do CBERS - satélite sino-brasileiro de recursos terrestres.** São José dos Campos. 2003-2006. Disponível em <http://www.cbers.inpe.br/pt/index_pt.htm>. Acesso em 15 de fevereiro de 2007.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Equatorial atmosphere research satellite home page.** São José dos Campos. 2005. Disponível em <http://www.laser.inpe.br/equars>. Acesso em 12 de janeiro de 2007.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Homepage do programa de satélites científicos do INPE**. São José dos Campos, 2006. Disponível em <http://www.cea.inpe.br/satelites.php>. Acesso em 20 de dezembro de 2006.

JOHNSTON, M. SPIKE: AI scheduling for NASA's Hubble space telescope. In: IEEE CONFERENCE ON AI APPLICATIONS (CAIA '90), 6., 1990, Santa Barbara, CA, USA. **Proceedings...** Santa Barbara: IEEE, 1990. pp. 184-190.

JOHNSTON, M.; MILLER, G. SPIKE: intelligent scheduling of Hubble space telescope observations. **Intelligent Scheduling,** M. Zweben, [ed.], Morgan Kaufman, pp. 391-422, 1994.

JONSSON, A.; MORRIS, P.; MUSCETTOLA, N.; RAJAN, K. Planning in interplanetary space: theory and practice. In: 2000 INTERNATIONAL CONFERENCE ON AI PLANNING AND SCHEDULING (AIPS), 2000, Breckenridge, CO, USA. **Proceedings...** [s.n.], 2000. pp. 177-186.

KAUTZ, H.; SELMAN, B. Planning as satisfiability. In: EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE (ECAI92), 10., 1992, Vienna, Austria. **Proceedings...** Chichester: John Wiley and Sons, 1992. pp. 359 – 363.

KAUTZ, H.; SELMAN, B. BLACKBOX: a new approach to the application of theorem proving to problem solving. In: 1998 INTERNATIONAL CONFERENCE ON AI PLANNING AND SCHEDULING (AIPS), 4., 1998, Pittsburgh, USA. **Proceedings...** [s.n.], 1998. pp. 58-60.

KAUTZ, H. Deconstructing planning as satisfiability. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-06), 21., 2006, Boston, MA, USA. **Proceedings...** Boston: AAAI Press, 2006.

KUCINSKIS, F. N.; FERREIRA, M. G. V. Dynamic allocation of resources to improve scientific return with onboard automated planning. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 9., 2006, Rome, Italy. **Proceedings...** Rome: AIAA, 2006.

KUCINSKIS, F. N.; FERREIRA, M. G. V.; ARIAS, R. Increasing the autonomy of scientific satellites to deal with short-duration phenomena. In: IEEE AEROSPACE CONFERENCE, 2007, Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2007.

LIU, D.; MCCLUSKEY, T. L. **The OCL language manual, version 1.2.** Huddersfield, UK. 2000. Technical Report, Department of Computing and Mathematical Sciences, University of Huddersfield.

MCALLESTER, D.; ROSENBLITT, D. Systematic nonlinear planning. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-91), 9., 1991, Anaheim, CA, USA. **Proceedings...** Anaheim: AAAI Press, 1991. pp. 634-639.

MCDERMOTT, D. **PDDL - the planning domain definition language.** Yale, CT, USA. 1998. Technical Report CVC TR-98-003/DCS, Yale Center for Computational Vision and Control.

MINTON, S.; KNOBLOCK, C.; KUOKKA, D. GIL; Y., JOSEPH, R.; CARBONELL, J. **Prodigy 2.0: the manual and tutorial.** Pittsburg, PA, USA, 1989. Technical Report CMU-CS-89-146, University of Carnegie Mellon.

MUSCETTOLA, N.; PELL, B.; HANSSON, O.; MOHAN, S. Automating mission scheduling for space-based observatories. In: **Robotic Telescopes: Current Capabilities, Present Developments, and Future Prospects for Automated Astronomy**. G.W. Henry and J.A. Eaton (eds.), Astronomical Society of the Pacific, Provo, UT, USA, 1995.

MUSCETTOLA, N.; DORAIS, G. A.; FRY, C.; LEVINSON, R.; PLAUNT, C. IDEA: planning at the core of autonomous reactive agents. In: 2002 INTERNATIONAL CONFERENCE ON AI PLANNING AND SCHEDULING (AIPS), 6., 2002, Toulouse, France. **Proceedings...** [s.n.], 2002. pp. 58-60.

MYERS, K. L.; SMITH, S. F. Issues in the integration of planning and scheduling for enterprise control. In: DARPA-JFACC SYMPOSIUM ON ADVANCES IN ENTERPRISE CONTROL, 1999, San Diego, CA, USA. **Proceedings...** [s.n.], 1999.

PAVLOVIC, V.; GARG, A.; KASIF, S. **Where we were and where to next.** Summary of the workshop on machine learning technologies for autonomous space applications. Washington, D.C., USA, August 2003. 20th International Conference on Machine Learning (ICML-2003).

PEDERSEN L.; SMITH D.; DEANS M.; SARGENT R.; KUNZ C.; LEES D.; RAJAGOPALAN S. Mission planning and target tracking for autonomous instrument placement. In: IEEE AEROSPACE CONFERENCE, 2005, Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2005.

PEDNAULT, E. ADL: exploring the middle ground between STRIPS and the situation calculus. In: INTERNATIONAL CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING (KR-89), 1., 1989, Toronto, Canada. **Proceedings...** San Francisco: Morgan Kaufmann Publishers Inc., 1989. pp. 324-332.

PENBERTHY, J. S.; WELD, D. S. UCPOP: A sound, complete, partial-order planner for ADL. In: INTERNATIONAL CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING (KR-92), 3., 1992, Cambridge, MA, USA. **Proceedings...** San Francisco: Morgan Kaufmann Publishers Inc., 1992.

RABIDEAU, G.; CHIEN, S.; MANN, T.; WILLIS, J.; SIEWERT, S.; STONE, P. Interactive, repair-based planning and scheduling for shuttle payload operations. In: IEEE AEROSPACE CONFERENCE, 1997, Aspen, CO, USA. **Proceedings...** Aspen: IEEE, 1997.

RABIDEAU, G.; KNIGHT, R.; CHIEN, S.; FUKUNAGA, A.; GOVINDJEE, A. Iterative repair planning for spacecraft operations using the ASPEN system. In: INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE ROBOTICS AND AUTOMATION IN SPACE (I-SAIRAS), 5., 1999, Noordwijk, The Netherlands. **Proceedings...** Noordwijk: ESA/ESTEC, 1999.

RUML, W.; FROMHERZ, M. P. J. On-line planning and scheduling for high-speed manufacturing. In: INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING (ICAPS04), 2004, Whisler, Canada. **Proceedings...** Whisler: AIAA, 2004.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. 2. ed., São Paulo: Editora Campus, 2004. 1040 p. ISBN (8535211772).

SACERDOTI, E., D. Planning in a hierarchy of abstraction spaces. **Artificial Intelligence**, v. 5, n. 2, pp. 115-135, 1974.

SILVA, F.; CASTILHO, M.; KÜNZLE, L. PETRIPLAN: a new algorithm for plan generation (preliminary report). In: INTERNATIONAL JOINT CONFERENCE (IBERAMIA/SBIA), IBERO-AMERICAN CONFERENCE ON AI, 7., BRAZILIAN SYMPOSIUM ON AI, 15., 2000, Atibaia, SP, Brasil. **Proceedings...** São Paulo: ICMS/USP, 2000. pp. 86–95.

SILVA, F. V. **Planejamento automático aplicado a problemas dependentes de recursos.** 2003. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal do Ceará, Fortaleza, 2003.

SIMMONS, R.; APFELBAUM, D. A task description language for robot control. In: Conference on Intelligent Robotics and Systems, 1998, Vancouver, Canada. **Proceedings...** [s.n.], 1998.

SMITH, D. E.; FRANK, J.; JONSSON, A. K. Bridging the gap between planning and scheduling. **Knowledge Engineering Review**, v. 15, n. 1, pp. 47-83, 2000.

TATE, A. Generating project networks. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI), 5., 1977, Cambridge, MA, USA. **Proceedings...** Cambridge: William Kaufmann, 1977. pp. 888–893.

TESTON, F.; CREASEY, R.; BERMYN, J.; MELLAB, K. PROBA: ESA's autonomy and technology demonstration mission. In: INTERNATIONAL ASTRONAUTICAL CONGRESS (IAC), 48., 1997, Turin, Italy. **Proceedings...**Turin: AIAA, 1997.

VERE, S. Planning in time: windows and durations for activities and goals. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 5, pp. 246-267, 1983.

VERMA, V.; JONSSON, A.; SIMMONS, R.; ESTLIN, T.; LEVINSON, R. Survey of command execution systems for NASA robots and spacecrafts. In: INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING (ICAPS05), 2005, Monterey, CA, USA. **Proceedings...** Monterey: AIAA, 2005. pp. 40–49.

WERTZ, J.; LARSON, W. (Ed.) **Space mission analysis and design.** 3. ed. Torrance, CA: Microcosm, Inc. and Kluwer Academic Publishers, 1999. 969 p. ISBN (978-1881883104).

ZWEBEN, B.; DAVIS, M.; DAUN, E.; DEALE, M. Scheduling and rescheduling with iterative repair. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 23, n. 6, Nov / Dec 1993.

APÊNDICE A

MODELO DO SATÉLITE E EXPERIMENTOS EM RASSO_ML

```
#include "language.h"
#include "config.h"


/* CRIACAO DOS DOMINIOS A SEREM UTILIZADOS NO MODELO ***************/

create_domain(Exp_Name,
                    certo, grom, ionex, mltm, tip);

create_domain(Mode,
                    full, partial);


/* CRIACAO DAS CLASSES DE OBJETOS DO MODELO ************************/

create_class(Antenna,
                    bool   on;
             );

create_class(Experiment,
                    Exp_Name      name;
                    bool          on;
                    Mode          mode;
                    int           sample_rate;
                    int           precision;
                    int           priority;
            );


/* CRIACAO DOS OBJETOS E RECURSOS DO MODELO ************************/

create_object(CERTO,       Experiment);
create_object(GROM,        Experiment);
create_object(IONEX,       Experiment);
create_object(MLTM,        Experiment);
create_object(TIP,         Experiment);
create_object(Comm_System, Antenna);

create_resource(Power);
create_resource(Mass_Memory);

resource_is_consumed_by(Power, CERTO);
resource_is_consumed_by(Power, GROM);
resource_is_consumed_by(Power, IONEX);
resource_is_consumed_by(Power, MLTM);
resource_is_consumed_by(Power, TIP);
resource_is_consumed_by(Power, Comm_System);

resource_is_consumed_by(Mass_Memory, CERTO);
resource_is_consumed_by(Mass_Memory, GROM);
resource_is_consumed_by(Mass_Memory, IONEX);
resource_is_consumed_by(Mass_Memory, MLTM);
resource_is_consumed_by(Mass_Memory, TIP);
```

```
/* DEFINICAO DAS ACOES DO MODELO ***********************************/

// "Turn_On" nao eh aplicavel apenas ao experimento IONEX
RASSO_Action Turn_On (action_parameters)
{
    parameter(exp);

    when_planning
    {
        condition(get_current_state_by_id(Experiment, exp).on == false);

        // ionex nao pode ser ligado; apenas seu modo de operacao alterado
        condition(get_current_state_by_id(Experiment, exp).name != ionex);

        force_state_if_needed_by_id(Experiment, exp, on, false);

        //efeitos descritos a partir daqui
        set_current_state_by_id(Experiment, exp).on = true;

        switch (get_current_state_by_id(Experiment, exp).name)
        {
            case certo:
            {
                consume_resource_by_id(Experiment, exp, Power, 3.6);
                consume_resource_by_id(Experiment, exp, Mass_Memory, 420.98
                                                                     per_min);
                break;
            }
            case grom:
            {
                consume_resource_by_id(Experiment, exp, Power, 12);
                consume_resource_by_id(Experiment, exp, Mass_Memory, 711.12
                                                                     per_min);
                break;
            }
            case mltm:
            {
                consume_resource_by_id(Experiment, exp, Power, 14);
                consume_resource_by_id(Experiment, exp, Mass_Memory, 264.54
                                                                     per_min);
                break;
            }
            case tip:
            {
                consume_resource_by_id(Experiment, exp, Power, 7);
                consume_resource_by_id(Experiment, exp, Mass_Memory, 568.89
                                                                     per_min);
                break;
            }
            default: break; // apenas para evitar warning
        }
    }

    when_running
    {
        printf("Turn_On experiment %i\n", exp);
    }

    action_success;
}
```

```
// "Turn_Off" nao eh aplicavel apenas ao experimento IONEX
RASSO_Action Turn_Off (action_parameters)
{
    parameter(exp);

    when_planning
    {
        condition(get_current_state_by_id(Experiment, exp).on == true);

        //ionex nao pode ser desligado; apenas seu modo de operacao alterado
        condition(get_current_state_by_id(Experiment, exp).name != ionex);

        force_state_if_needed_by_id(Experiment, exp, on, true);

        //efeitos descritos a partir daqui
        set_current_state_by_id(Experiment, exp).on = false;

        consume_resource_by_id(Experiment, exp, Power, 0);
        consume_resource_by_id(Experiment, exp, Mass_Memory, 0);
    }

    when_running
    {
        printf("Turn_Off experiment %i\n", exp);
    }

    action_success;
}

// Aplicavel apenas ao IONEX
RASSO_Action Change_Mode (action_parameters)
{
    parameter(exp);
    parameter(new_mode);

    when_planning
    {
        condition(get_current_state_by_id(Experiment, exp).name == ionex);
        condition(get_current_state(IONEX).on == true);
        condition(get_current_state(IONEX).mode != new_mode);

        force_state_if_needed(IONEX, on, true);

        //efeitos descritos a partir daqui
        set_current_state(IONEX).mode = new_mode;

        if (new_mode == full)
        {
            consume_resource_by_id(Experiment, exp, Power, 5.4);
            consume_resource_by_id(Experiment, exp, Mass_Memory, 189.16
                                                                per_min);
        }
        else //new_mode == partial
        {
            consume_resource_by_id(Experiment, exp, Power, 3);
            consume_resource_by_id(Experiment, exp, Mass_Memory, 95.29
                                                                per_min);
        }
    }

    when_running
    {
        printf("Changing IONEX to mode %i\n", new_mode);
```

```
    }

    action_success;
}

// Aplicavel apenas ao CERTO e MLTM.
// Recebe new_sr como valores discretos: 1, 2, ou 3
RASSO_Action Change_Sample_Rate (action_parameters)
{
    parameter(exp);
    parameter(new_sr);

    double new_power = 0, new_memory = 0;

    when_planning
    {
        condition(get_current_state_by_id(Experiment, exp).on == true);
        condition(get_current_state_by_id(Experiment, exp).sample_rate !=
                                                              new_sr);

        force_state_if_needed_by_id(Experiment, exp, on, true);

        // efeitos descritos a partir daqui
        set_current_state_by_id(Experiment, exp).sample_rate = new_sr;

        switch (get_current_state_by_id(Experiment, exp).name)
        {
            case certo:
            {
                new_power  = 3.6    * new_sr;
                new_memory = 420.98 * new_sr;
                break;
            }
            case grom:
            {
                new_power  = 12     * new_sr;
                new_memory = 711.12 * new_sr;
                break;
            }
            case ionex:
            {
                if (get_current_state(IONEX).mode == full)
                {
                    new_power  = 5.4    * new_sr;
                    new_memory = 189.16 * new_sr;
                }
                else // modo partial
                {
                    new_power  = 3      * new_sr;
                    new_memory = 95.29  * new_sr;
                }
                break;
            }
            case mltm:
            {
                new_power = 14      * new_sr;
                new_memory = 264.54 * new_sr;
                break;
            }
            case tip:
            {
                new_power = 7       * new_sr;
                new_memory = 568.89 * new_sr;
```

```
                break;
            }
        }

        consume_resource_by_id(Experiment, exp, Power, new_power);
        consume_resource_by_id(Experiment, exp, Mass_Memory, new_memory
                                                              per_min);
    }

    when_running
    {
        printf("Changing exp. %i sample_rate to %i\n", exp, new_sr);
    }

    action_success;
}

// Aplicavel ao CERTO E GROM
// Recebe new_precision como valores discretos: 1, 2, ou 3
RASSO_Action Change_Precision (action_parameters)
{
    parameter(exp);
    parameter(new_precision);

    double new_memory = 0;

    when_planning
    {
        condition(get_current_state_by_id(Experiment, exp).on == true);
        condition(get_current_state_by_id(Experiment, exp).precision !=
                                                        new_precision);

        condition((get_current_state_by_id(Experiment, exp).name == certo) ||
                  (get_current_state_by_id(Experiment, exp).name == grom));

        force_state_if_needed_by_id(Experiment, exp, on, true);

        // efeitos descritos a partir daqui
        set_current_state_by_id(Experiment, exp).precision = new_precision;

        switch (get_current_state_by_id(Experiment, exp).name)
        {
            case certo:
            {
                new_memory = 420.98 * new_precision;
                break;
            }
            case grom:
            {
                new_memory = 711.12 * new_precision;
                break;
            }
            default: break; // apenas para nao disparar warning
        }

        consume_resource_by_id(Experiment, exp, Mass_Memory, new_memory
                                                              per_min);
    }

    when_running
    {
        printf("Changing precision to %i\n", new_precision);
    }
```

```
    action_success;
}


/* DEFINICAO DAS JANELAS DE TEMPO E COMPORTAMENTOS DO MODELO **************/

create_time_window(Day);
create_time_window(Night);
create_time_window(Communicating);

RASSO_Behavior When_Day (behavior_parameters)
{
    at_start
    {
        call_action(Turn_On, mltm);

        // garante de 264.54 a 300 kbits/m p/MLTM nos primeiros 15 m da janela
        guarantee_resource(MLTM, Mass_Memory, 264.54 per_min, 300 per_min, 0,
                                                                   15 * 60);
    }

    at_end
    {
        call_action(Turn_Off, mltm);
    }
}

RASSO_Behavior When_Night (behavior_parameters)
{
    at_start
    {
         // garante 5.4Watts p/IONEX (modo full) nos primeiros 12 m da janela
        guarantee_resource(IONEX, Power, 5.4, 0, 0, 12 * 60);

        call_action(Turn_On, certo);
    }

    at_end
    {
        call_action(Turn_Off, certo);
    }
}

RASSO_Behavior When_Communicating (behavior_parameters)
{
    at_start // taxa de transferencia para solo: 230 kbits/s
    {
        set_current_state(Comm_System).on = true; // liga a antena

        // libera 13.8 Mbits por minuto da memoria de massa
        generate_resource(Mass_Memory, 13800 per_min);

        // o sistema de comunicacao consome 10W
        consume_resource(Comm_System, Power, 10);
    }

    at_end
    {
        // para de liberar memoria e de consumir energia
        generate_resource(Mass_Memory, 0);
        consume_resource(Comm_System, Power, 0);
```

```
        set_current_state(Comm_System).on = false; // desliga a antena
    }
}


/* INICIALIZACAO DO MODELO ***********************************************/

void Initialize_Model()
{
    // Inicializa o controle de classes
    initialize_class(Experiment);
    initialize_class(Antenna);

    initialize_object(GROM, Experiment);
    initialize_object(MLTM, Experiment);

    // a bateria pode fornecer 50W quando plena
    initialize_resource(Power, reservable, 0, 50);

    // 120 Mbits dedicados aos experimentos
    initialize_resource(Mass_Memory, depletable, 0, 120 * 1024);

    initialize_time_window(Day);
    initialize_time_window(Night);
    initialize_time_window(Communicating);

    behavior_occurs_at_time_window(When_Day, Day);
    behavior_occurs_at_time_window(When_Night, Night);
    behavior_occurs_at_time_window(When_Communicating, Day);
    behavior_occurs_at_time_window(When_Communicating, Night);

    resource_is_consumed_by(Power, GROM);
    resource_is_consumed_by(Power, MLTM);

    resource_is_consumed_by(Mass_Memory, GROM);
    resource_is_consumed_by(Mass_Memory, MLTM);
}
```

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto executáveis.